Spring 2009

TREE-D-SEEK: A Framework for Retrieving Three-Dimensional Scenes

Saurav Mazumdar
Old Dominion University

TREE-D-SEEK: A FRAMEWORK FOR RETRIEVING 3D SCENES

by

Saurav Mazumdar
MS. December 2002, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

ELECTRICAL AND COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
May 2009

Approved by:

Lee A. Belfore II (Director)

Frederic D. McKenzie (Member)

Roland R. Mielke (Member)

Andreas Tolk (Member)

ABSTRACT

TREE-D-SEEK: A FRAMEWORK FOR RETRIEVING 3D SCENES

Saurav Mazumdar
Old Dominion University, 2009
Director: Dr Lee A Belfore, II.

In this dissertation, a strategy and framework for retrieving 3D scenes is proposed. The strategy is to retrieve 3D scenes based on a unified approach for indexing content from disparate information sources and information levels. The TREE-D-SEEK framework implements the proposed strategy for retrieving 3D scenes and is capable of indexing content from a variety of corpora at distinct information levels. A semantic annotation model for indexing 3D scenes in the TREE-D-SEEK framework is also proposed. The semantic annotation model is based on an ontology for rapid prototyping of 3D virtual worlds.

With ongoing improvements in computer hardware and 3D technology, the cost associated with the acquisition, production and deployment of 3D scenes is decreasing. As a consequence, there is a need for efficient 3D retrieval systems for the increasing number of 3D scenes in corpora. An efficient 3D retrieval system provides several benefits such as enhanced sharing and reuse of 3D scenes and 3D content. Existing 3D retrieval systems are closed systems and provide search solutions based on a predefined set of indexing and matching algorithms. Existing 3D search systems and search solutions cannot be customized for specific requirements, type of information source and information level.

In this research, TREE-D-SEEK- an open, extensible framework for retrieving 3D scenes- is proposed. The TREE-D-SEEK framework is capable of retrieving 3D scenes based on indexing low level content to high-level semantic metadata. The TREE-D-SEEK framework is discussed from a software architecture perspective. The architecture is based on a common process flow derived from indexing disparate information sources. Several indexing and matching algorithms are implemented. Experiments are conducted to evaluate the usability and performance of the framework. Retrieval performance of the framework is evaluated using benchmarks and manually collected corpora.

A generic, semantic annotation model is proposed for indexing a 3D scene. The primary objective of using the semantic annotation model in the TREE-D-SEEK framework is to improve retrieval relevance and to support richer queries within a 3D scene. The semantic annotation model is driven by an ontology. The ontology is derived from a 3D rapid prototyping framework. The TREE-D-SEEK framework supports querying by example, keyword based and semantic annotation based query types for retrieving 3D scenes.

This dissertation is dedicated to my father Sachish Chandra Mazumdar.

ACKNOWLEDGEMENTS

"Don Quixote: Dost not see? A monstrous giant of infamous repute whom I intend to encounter.
Sancho Panza: It's a windmill.
Don Quixote: A giant. Canst thou not see the four great arms whirling at his back?
Sancho Panza: A giant?
Don Quixote: Exactly."[1]

Indeed, when I began this journey the distinction between windmills, giants and doctoral dissertations were fuzzy to say the least. At the end of this journey, the distinctions have become fuzzier and irrelevant. The journey itself was full of interesting non-windmill sights, events and interactions, and I have several co-travelers and guides to thank for making the journey's end a success.

My advisor, Dr Lee Belfore provided the initial research idea that germinated into this work. Dr Belfore not only provided valuable continuous feedback on the research front but also created an environment of support and trust on a personal level that is crucial for an international graduate student embarking on this journey. My committee members, Dr Andreas Tolk, Dr Rick Mckenzie and Dr Roland Mielke provided excellent feedback and helped shape the final outcome. My master's thesis advisor, Dr James Leathrum got me to love research and more importantly on how to aspire to hit a three iron. Dr Sacharia Albin, the program director, Dr Shirshak Dhali, the chair of our department and Dr Zahorian, the former chair of our department, helped in ensuring that my progress was

---

[1] Man of La Mancha (1972)

steady and the journey's end achievable. My colleagues, Dr Emre Baydogan and Prabhu Krishnan provided the initial steps with their excellent research work with the Common Scene Definition Framework. Prabhu also provided financial and emotional support during some difficult times. My friend Ajay Kongera provided refuge, television and high speed internet during my endless trips to Norfolk. He also steadfastly provided much needed logistical support in my battle to submit the dissertation on time. Linda Marshall and Romina Samson from the ECE department helped tremendously in keeping the administrative paper work as fun and light as possible. Dr Yuri Millo, director of Simulation and Training Environment Lab (SiTEL) and everyone at SiTEL were very supportive and helped by letting me take some time off during my internship to finish this work.

I owe everything to my parents and family. My mother, Srimoti Mazumdar has been a source of inspiration and comfort all through this work. My regular conversations with her provided a much needed level of abstraction from the vagaries of research. My sister, Dr Sudeshna Mazumdar Leighton and my brother in law Dr Denys Leighton provided constant encouragement and support throughout this dissertation. My in-laws, Bhaskar and Bhama Menon provided much needed support during the last stages of this work.

An important motivator who has to be mentioned and who provided all the joy, entertainment and the urgency to finish - my ten month old daughter Sitara Mazumdar. Finally and most importantly, I could not have gone on this journey without my co-traveler, my GPS, my wife Ishita.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Chapter I

INTRODUCTION

Finding efficient ways of retrieving media content is an area of active research. In comparison to work related to retrieving text, audio, images, and video, research in retrieving 3D content is relatively recent. With ongoing improvements in computer hardware, computer networking and 3D authoring tools, the acquisition, production and deployment of 3D objects/3D scenes is become easier. 3D objects and scenes have found applications in many areas such as in biomedical applications, gaming, social networking, electronic commerce, security, etc.

In the field of medicine, the use of 3D models for human and animal anatomy is growing rapidly. These 3D models may be part of a patient's health record. Effective retrieval of 3D anatomical models would enhance health care processes and create an effective platform for knowledge sharing between any concerned parties in the health care process chain. Effective storage and retrieval mechanisms could enhance existing web-based public heath record systems such as Microsoft Health Vault [1] or Google Health [2]. An effective retrieval system for 3D medical models also promotes dissemination and training for educational purposes. Content based searches may be useful for detecting any anomalies in the 3D models present in the system.

Reference Model: IEEE Transactions on Computers.

In Bioinformatics, 3D shape matching is important in several areas such as for classifying proteins. In 3D gaming and simulation, an effective search capability promotes reuse of 3D assets. Indeed, 3D social networks such as *Second Life* [3] are becoming very popular. In large scale virtual worlds, search is crucial not just for locating objects but also for navigation within the virtual world. 3D models have also become very important for e-commerce applications. 3D models of products provide improved visualization of the commodity being retailed. 3D content based retrieval may be useful for face recognition systems and for detecting dangerous situations. In computer-aided manufacturing, an efficient 3D search and retrieval system is crucial for re-use and analysis. Based on the above discussion, it can be safely assumed that a significant increase in the number and usage of 3D models and repositories is an eventuality.

Therefore, there is a need for an efficient retrieval system for 3D content. For the purposes of this dissertation, 3D content refers to 3D virtual scenes and 3D objects present in 3D virtual scenes. A 3D virtual scene is a simulated 3D environment wherein the user/participant can interact with the objects in the environment or the environment itself. Retrieving 3D content involves indexing the 3D content, formulating a query, comparing the query to the 3D content, and returning the results or hits to the user. A user of the retrieval system refers to the human who submits the query to the retrieval system. Indexing refers to the process of collecting, parsing, extracting or detecting features, creating descriptors and storing the descriptors in a structure for fast retrieval. A feature is a characteristic of the data in a corpus. For instance, shape of a 3D object may be

deemed a feature and used for indexing. A descriptor provides syntax and an objective value for features so that they can be stored and evaluated in a consistent manner. Matching refers to the process of computing a similarity measure so that pairs of descriptors can be compared and scored. A few existing 3D model search engines and available 3D repositories are reviewed next.

1.1 Currently available 3D model search engines and search engines for the semantic web

Retrieval of 3D content is an area of active research interest. Several experimental 3D search engines are available both on the World Wide Web (WWW) and offline. In this section, a brief overview of the Princeton 3D model search engine [4], 3DESS [5], Google 3D warehouse [6], Ogden VI [7], ITI 3D search [8], SWOOGLE [9] is provided.

1.1.1 Princeton 3D model search engine

The Princeton 3D model search engine allows querying via text, 2D sketches and by uploading entire models. It uses rotational invariant spherical harmonics to compute shape descriptors. It is a complete search system with its own 3D focused crawler for WWW. The Princeton Group has also provided a benchmark for mesh models[4]. The query interface of the Princeton 3D model search engine is shown in Figure 1. The Princeton search engine supports multimodal queries that include combinations of text/2D sketch and text/3D sketch. No semantic indexing or querying is supported.

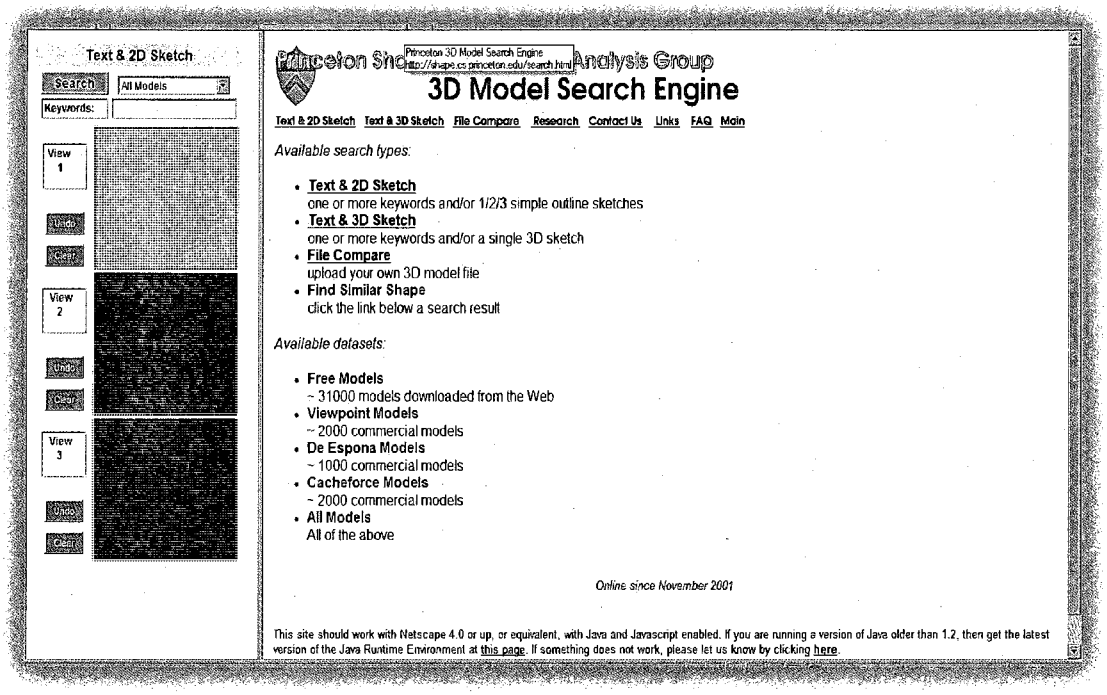


Figure 1: Princeton 3D model search engine query interface [4].

1.1.2 3-Dimensional Engineering Shape Search System (3DESS)

The 3DESS system [5] is a prototype shape search system for CAD/manufacturing 3D models. The system is not available online. It allows the user the choice of which type of shape based feature to use in the search. The search engine uses a relevance feedback approach to refine searching. No semantic indexing or querying is supported.

1.1.3 Google 3D warehouse

The Google 3D warehouse [6] is an online repository of 3D models. The 3D models are broadly classified into geo-referenced and non-geo-referenced. A geo-referenced 3D model is a real world object such as a stadium, building, etc. that can be accurately located using Google Earth. Non-geo-referenced models are other objects that are not

location specific such as a human body, plant, etc. The Google 3D warehouse query interface is shown in Figure 2. No shape indexing and matching is supported.

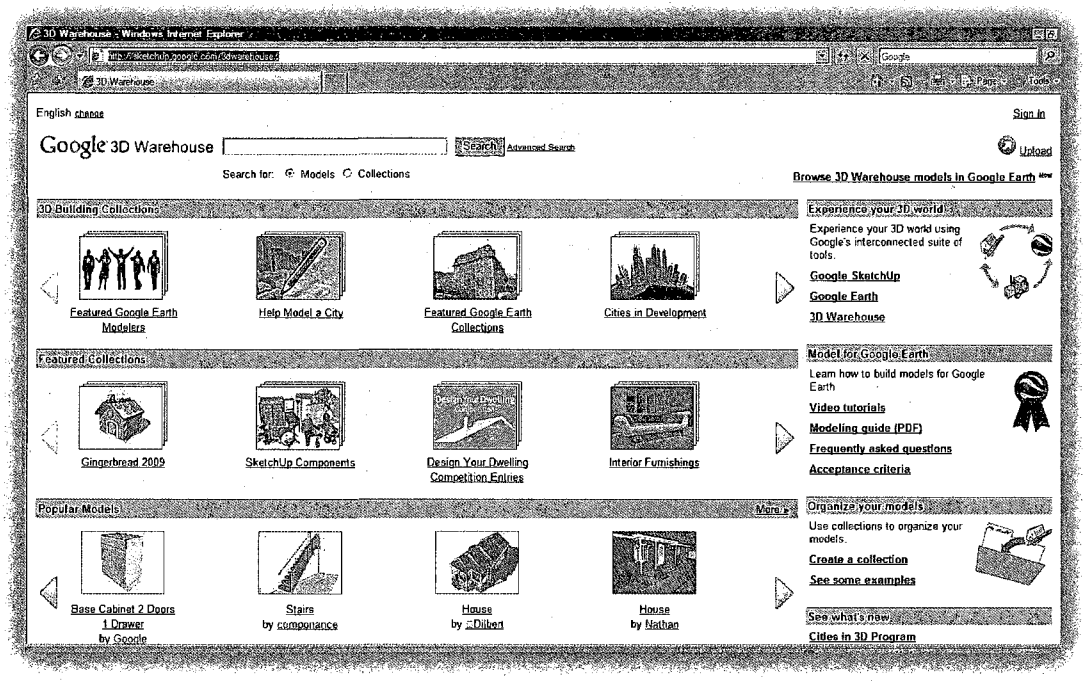


Figure 2: Google 3D warehouse query interface [6].

1.1.4 Ogden VI

The Ogden [7] VI is a search engine system that retrieves 3D content using rotational invariant shape descriptors, voxelized 3D models, point clouds, etc. Retrieval using similar parts of 3D models is also provided. The query interface of OGDEN VI is shown in Figure 3.

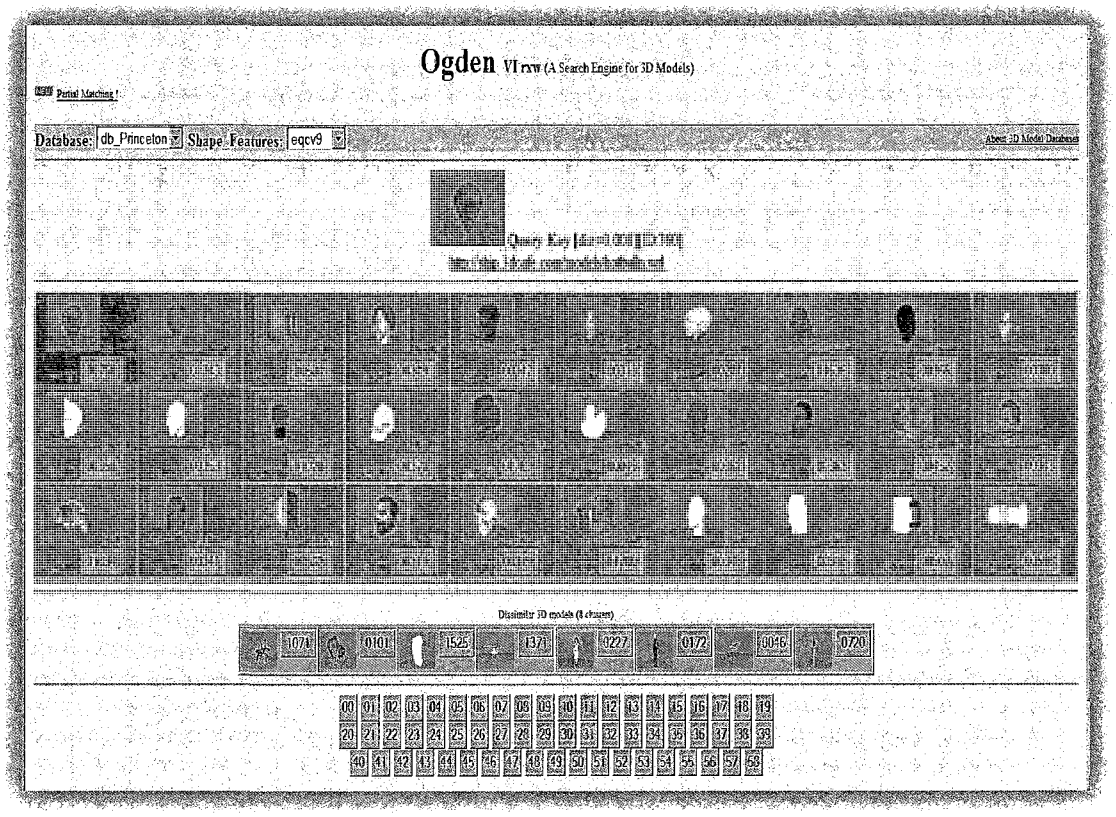


Figure 3: Ogden VI query interface [7].

1.1.5 Informatics and Telematics Institute (ITI) 3D search system

The ITI 3D search system [8] allows content based retrieval of 3D objects in Virtual Reality Markup Language (VRML). It allows searches on the Princeton Benchmark, the Utrecht database and its own database only. It does not support semantic indexing. The query interface is shown in Figure 4.

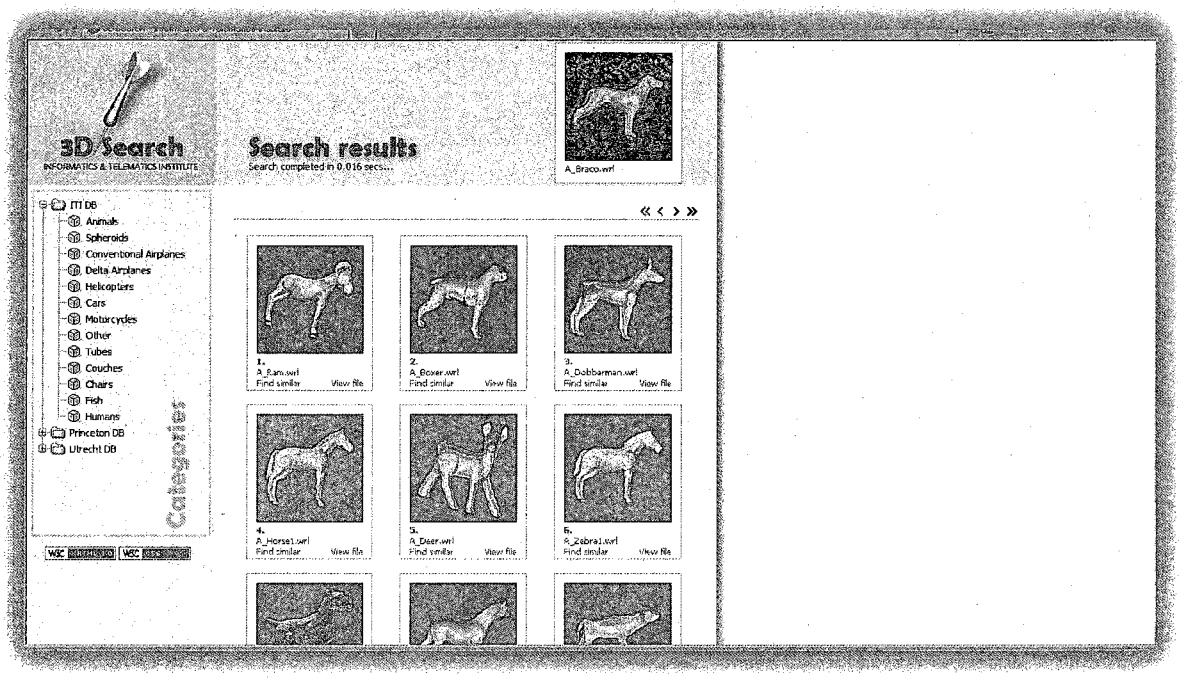


Figure 4: ITI 3D query interface [8].

1.1.5 SWOOGLE

The SWOOGLE system [9] is a complete search system for the semantic web. It is capable of crawling and indexing documents written in Web Ontology Language (OWL) and Resource Description Format (RDF) available on the web. It is not capable of indexing based on 3D low level content. The query interface is shown in Figure 5.

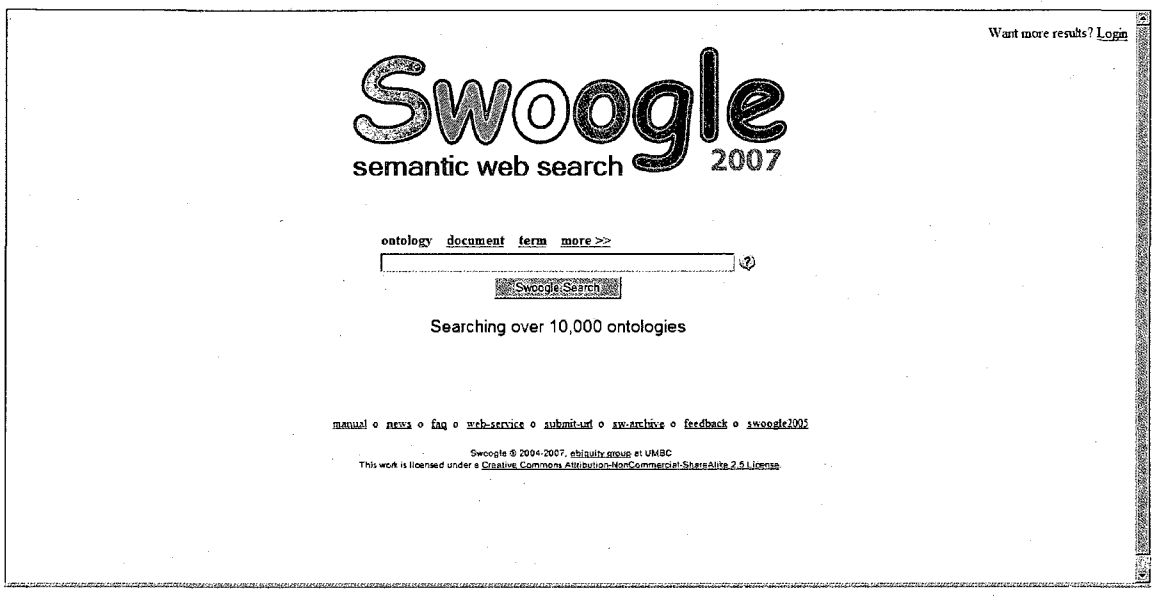


Figure 5: SWOOGLE query interface [9].

1.1.6 Discussion

Based on the survey of the above mentioned search engines, query interfaces and relevant work discussed in Chapter II, current retrieval solutions do not provide a unified approach of retrieving 3D scenes based on indexing and matching low level to high level content. An open framework for retrieving 3D scenes that can be used by search engine developers to create customized search solutions for 3D corpora has not been proposed or implemented. The problem statement is discussed next.

1.2 Problem statement

As the number of 3D models and scenes increase, there is a need for efficient 3D retrieval systems. The focus of several research works [4], [8], [5], [7] has been on identifying efficient low-level content-based retrieval mechanisms for 3D models. The primary goal of these research works was to identify effective 3D retrieval strategies for 3D models. The retrieval strategies proposed in these research works used either geometric or

statistical algorithms for indexing and matching 3D models. Several existing works and search engines [10] [6] used syntactic metadata and keyword based approaches for retrieving 3D content. The need for machine interpretability, richer querying capability, improved retrieval relevance and improved navigation in 3D scenes has resulted in several works [11][12] wherein semantic annotations were used to formally specify 3D scenes. Similar to retrieval in other types of multimedia, the "bridging" of the semantic gap from low level features to high level semantic metadata has been proposed.

In addition, in [13] and the above mentioned search systems, suitable modes of query for effective retrieval of 3D models were also investigated. The query mode supported by a search system is closely related to the choice of indexing and matching algorithm available in a search system. The general solution in available 3D search systems was to provide support for querying based on metadata, 2D/3D sketches and by example. In 3D retrieval based on 2D/3D sketches, a user provides a "skeleton" sketch of a 3D model and the same is matched against the contents of the 3D corpus. In querying by example, an entire 3D model is provided as query and similar 3D models are retrieved based on low level content matching. In addition, semantic queries based on natural language have been proposed for 3D scene retrieval. Existing search systems support a few or all of the above query modes. These search systems may also support retrieval based on multimodal queries wherein a combination of the above mentioned queries are used to retrieve 3D content. For example, in [10], a user is allowed to provide both text and 2D sketch as input query.

Based on the survey performed in 3D retrieval and 3D search systems, several observations may be made. First, a unifying retrieval strategy and an open framework for retrieving 3D scenes capable of indexing low-level content to high-level semantic metadata has not been proposed or implemented. Indeed, current solutions for retrieving 3D content employ a "stove-piped" approach of indexing and matching 3D content wherein at most, the retrieval system supports implementation and evaluation of algorithms at a particular information level and by individual research groups.

Second, a framework for retrieving 3D scenes has not been discussed. The above mentioned search and retrieval systems are closed systems and are used by particular research groups and organization for their own specific requirements or to support general users for performing 3D retrieval based on a predetermined set of retrieval mechanisms. The capability of creating a search engine for retrieving 3D scenes with customized indexing and matching algorithms for specific corpora has not been addressed.

Third, a framework that is both capable of retrieving 3D scenes and providing search within a 3D scene for navigation and for retrieving 3D objects within a scene has not been discussed. Existing retrieval solutions are capable of either retrieving 3D scenes that contain at most one 3D model or provide search for particular closed 3D scenes/virtual worlds.

Fourth, a framework that can index 3D scenes based on scenegraph content and structure has not been proposed. A scenegraph is a data structure that contains information about 3D content present in a scene. A scenegraph provides the capability of logically ordering 3D content present in a scene.

In this research, TREE-D-SEEK, a retrieval framework for 3D scenes, is proposed. The TREE-D-SEEK framework implements a unified strategy for indexing and matching 3D content. The TREE-D-SEEK framework is capable of indexing and matching 3D content based on textual annotations, scenegraph content and structure, shape and semantic annotations. Based on the individual process flow in retrieving 3D content from each type of information source and information level, a unified process flow is derived. Indexing a 3D scene based on its scenegraph content and structure provides the capability of retrieving occluded or cluttered 3D scenes.

The software architecture of the TREE-D-SEEK framework is specified. Clear software interfaces are available to provide extensibility with respect to implementing a desired 3D retrieval strategy at the information levels supported in the retrieval strategy. As proof of concept, several indexing and matching algorithms have been implemented. Finally, an ontology based annotation model for indexing 3D scenes is proposed in this research. It is envisioned that this annotation model may be used to author 3D scenes and to support richer queries for improved navigation and search within a 3D scene.

1.3 Research scope

One of the primary objectives of this research is to create a generic 3D retrieval strategy and a framework that implements this strategy for retrieving 3D scenes design. For the purposes of this research, 3D scene content refers to 3D objects present in a virtual scene. The goal of this research is not in designing the "front-ends" vis-à-vis the user interface for querying or for viewing the results. Also, retrieving 3D content based on the behavior of objects present in the scene is not supported.

The TREE-D-SEEK framework must contain clear software interfaces so that it can potentially be part of a 3D authoring tool that may a 3D search capability. The framework should allow seamless integration of indexing and matching techniques for the supported information sources and information levels. A semantic annotation model must be proposed for indexing any generic 3D scene.

1.4 Chapter organization

The rest of the dissertation is organized as follows.

Chapter 2: Background and Related Work: In this chapter, background concepts and related work in 3D retrieval are surveyed. Several indexing and matching algorithms are discussed.

Chapter 3: TREE-D-SEEK Retrieval Strategy. In this chapter, a generic retrieval strategy for retrieving 3D scenes is proposed. A common process flow is derived based on existing process flows for retrieval based on metadata, scenegraph, shape and semantic annotations are discussed.

Chapter 4: TREE-D-SEEK Framework. The TREE-D-SEEK framework is proposed in Chapter 3. The software architecture of the framework is described. Several indexing and matching algorithms implemented in the TREE-D-SEEK framework are discussed in this chapter.

Chapter 5: TREE-D-SEEK Semantic Annotation Model. In this chapter, a semantic annotation model for 3D scenes is proposed. The annotation model is based on an ontology derived from a 3D rapid prototyping framework.

Chapter 6: Implementation, Experiments, and Discussion. In this chapter, the TREE-D-SEEK framework is used on selected 3D corpora and several retrieval performance assessments are presented.

Chapter 7: Conclusions and Future Work. This chapter describes the conclusions drawn from this research. Future enhancements that may be potentially pursued are also outlined.

Chapter II

BACKGROUND AND RELATED WORK

The primary objective of this dissertation is to create an Information Retrieval (IR) framework capable of retrieving 3D scenes based on extracting low level features to high level semantic metadata. In Chapter I, several complete search systems for 3D content and semantic metadata were discussed. In this chapter, background and related work pertaining to available indexing and matching techniques are discussed.

In particular, a brief discussion of 3D model representation and two 3D file formats is first presented. This is followed by a description of scenegraphs and scenegraph-based 3D technologies. Next, relevant background and work in indexing, matching and retrieving content using text-based, structure-based, content-based and semantic metadata is described.

For text-based retrieval, a brief discussion of vector based method is provided. For structure-based retrieval, a description of related work done in software versioning systems and XML versioning systems is provided. For semantic metadata-based retrieval, firstly existing standards available for multimedia and 3D retrieval are discussed. Subsequently, for semantic metadata based retrieval, a brief description of semantic retrieval based on ontologies for 3D shapes is mentioned. Finally, a discussion and summary of this chapter is provided.

2.1 3D model representation

A 3D scene may be comprised of one or more 3D models/objects. 3D models representations can vary in several ways. For instance, 3D object representations can vary with respect to the ease in acquisition, storage (size), rendering efficiency, authoring efficiency, articulated transformational capability, etc. 3D objects representation may be classified as shown in Figure 6 [14] . 3D objects may be broadly represented by

- Points
- Surfaces
- Solids
- High level

Point representation refers to an unordered, raw representation of 3D models. Examples of Point representations are point clouds, range images and polygonal soups. Point clouds are simply a set of 3D vertices. Point clouds may be obtained using 3D scanners. Range images are also known as depth maps. Pixels in a range image represent the distance of a point in the scene from a reference frame. By using a range of views/reference frames, the 3D shape can be estimated and reconstructed. Range images may be acquired using range scanners. Polygonal soups are an unordered collection of polygons. Polygonal soup representation of 3D objects does not have any information relating to how the polygons are interconnected with each other. Most graphic cards can support polygonal soup representations. It is quite common to find hardware support for triangles as a primitive for rendering 3D models.

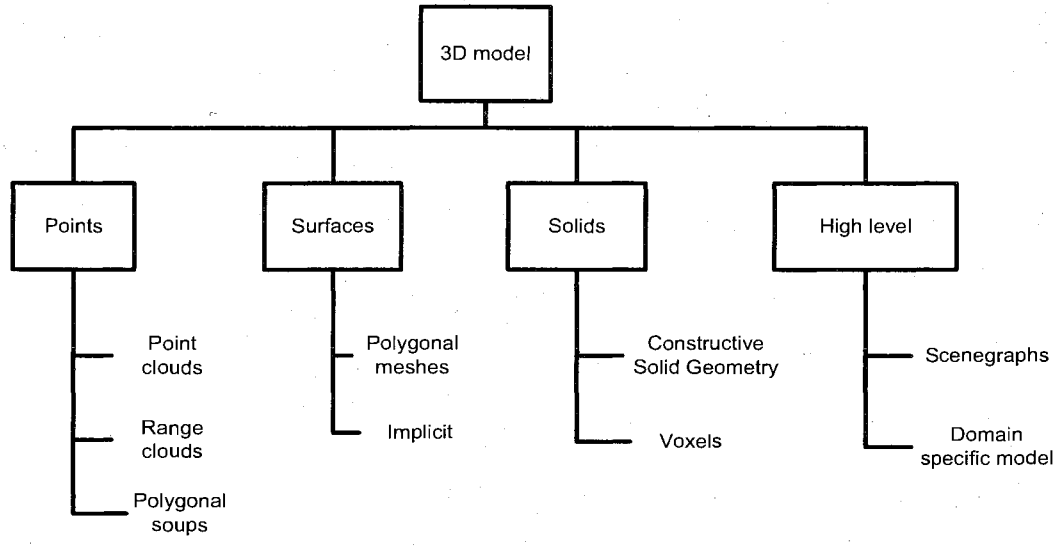


Figure 6: 3D model representation.

Surface information of a 3D object is used in 3D surface object representations. Examples of surface representations are polygonal meshes and parametric surface representations. In polygonal meshes representation, an order for specifying the vertices, edges and interconnectivity of polygons is provided. A vertex is shared by at least two edges, and each edge is shared at most by two polygons. Each polygon consists of a closed set of edges [15]. An implicit surface representation uses the implicit function $S = F(x, y, z) = 0$. By using this function, a 3D point can be easily determined to be inside, outside or on the surface.

Solid model representation refers to 3D objects that can be represented using rigid solids. The Solid model representation is also referred to as a Volume model. Examples of 3D solid representations are Constructive Solid Geometry (CSG) [14] and Voxels. CSGs use geometric primitives such as cylinders, spheres and boxes. The CSG objects are typically

rendered using set-theoretic functions such as unions and intersections on CSG primitives. The volumetric pixel or voxel is typically acquired using devices such as Magnetic Resonance Imaging (MRI), CAT (Computerized Axial Tomography), etc. The voxelizing process creates a uniform 3D grid that provides samples in three dimensions of the object being modeled. Initially, a volumetric dataset is constructed using a series of cross sectional images of the 3D object. Taking distance between each pixel (*interpixel distance*) within an image slice and in between each slice itself (*interslice distance*), a grid can be created. By interpolating interslice data, an entire volume can be represented [16].

High level 3D representations support easier authoring and rendering capabilities by using specialized data structures. Examples of high level 3D representations include 3D models for specific domains such as 3D protein modeling and scenegraphs. A scenegraph is a hierarchical data structure wherein each node of the data structure can contain some aspect of the 3D model or scene such as geometry of the object, associated transformation etc. Scenegraphs provide several benefits such as the ability to define objects in their own coordinate system, reuse of object definitions and articulated animation [17].

2.2 3D formats

A 3D format refers to how a 3D representation can be encoded for storage. One of the challenges of 3D retrieval relates to the availability of a wide variety of available 3D file

formats, both proprietary and non-proprietary file formats. Consequently, two non-proprietary formats are discussed briefly.

2.2.1 Virtual Reality Markup Language (VRML)

VRML is a technology for delivering 3D content over the web [18]. The VRML file is usually referred to as a VRML world and has an extension "wrl". VRML is a scene-graph based technology. A brief description of scenegraphs is provided later in this chapter. VRML provides several primitives called *nodes*. Nodes can be shape-related such as *Box* nodes, *Cylinder* nodes, *IndexedFaceSet* nodes, etc. In addition, nodes can be used to group together simpler nodes, i.e. they can be parent nodes that hold children nodes. Examples of parent nodes are *group* and *transform* nodes. Group and transform nodes allow composite objects composed of several simpler nodes to be controlled as one node. This provides the capability for specialized operations such as articulated animations, etc. Sensor nodes such as *TouchSensors* provide the capability to sense interactions between users and the associated nodes. Script nodes in VRML allow the use of ECMA-script or Java classes to add additional behavior for 3D scenes. Event routing is done using the keyword ROUTE. VRML has two keywords DEF and USE that allow object reuse. The *WorldInfo* node may be used for documentation and can appear anywhere in the file.

2.2.2 Extensible 3D (X3D)

Extensible 3D Graphics (X3D) is an ISO standard [19] for delivering 3D content with multimedia on a network. X3D was developed by the Web3D consortium. X3D provided the benefits of XML to VRML 2.0. The overall design criteria of X3D were to facilitate interchange and interoperability of 3D models by providing a common subset of 3D

techniques and graphic capabilities to map from and to various 3D software packages. X3D supports 2D/3D graphics, animations, scripting for complex behavior, user interactions, navigation, networking, audio/video etc.

X3D supports a scenegraph based architecture. A brief description of scenegraphs is available in the following section. The nodes of the scenegraph may be extended to add functionality for a particular requirement. Each node except the root node has a single parent. X3D allows for geometry rendering and expressing behavior. External scripting using Java script (ECMAScript) and Java is recognized. X3D browsers may be web browser based or standalone and may be used to render X3D scenes and allow animation and interaction. These browsers consist of a parser to read the file format and a scenegraph manager that recursively uses a depth-first-traversal to rapidly render the scene graphic nodes[20]. The Scene Access Interface (SAI) may be used to express complex behavior in X3D Scenes. The SAI may be accessed internally in the scenegraph via script nodes or externally from other programming languages (Java or JavaScript). This approach is different from VRML 2.0 as VRML 2.0 had two programming interfaces to express behavior.

To target specific 3D platforms and markets, the X3D specification supports *profiles*. A profile is a subset of functionalities provided in the X3D specification. This partitioning of functionalities into particular sets supports more efficient deployment of X3D scenes and worlds. There are six types of profiles available on X3D. The core profile is the simplest X3D profile and does not contain any geometry. The X3D core profile primarily

contains metadata nodes. This profile does not contain any animation capabilities. The *Interchange* profile contains all of the basic geometry nodes, and animation. This profile supports importing and exporting of 3D scenes. The *MPEG-4 Interactive* profile provides functionality required to specify 3D graphics in MPEG-4. The *CADInterchange* profile provides support in importing CAD models. It contains some CAD specific nodes. The *Immersive* profile incorporates all functionality available in VRML 2.0. The *Full* profile includes all the nodes in X3D. It contains capabilities such as Humanoid Animation (H-Anim), Non Uniform Rational B-spline Surfaces (NURBS).

Each X3D profile consists of a collection of *components*. There are twenty four components in X3D 3.0 [20]. Each component contains a set of specific X3D nodes. An X3D developer can import desired functionality at the component level from any profile. This allows the X3D developer to select specific functionality from any desired profile without having to import the entire profile.

2.2.2.1 X3D file structure

The X3D file structure is shown in Figure 7.

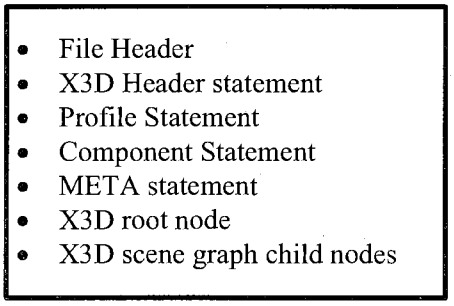


Figure 7: X3D file structure.

The file header indicates that the file is an XML one and also specifies the text encoding. The X3D header statement contains the schema association and namespace for X3D. The profile statement indicates the type of profile the X3D world would use. The Component statement provides an additional level of usability wherein individual components not belonging to the above specified profile in the file may be imported and used. The Meta statements may be used to specify the metadata associated with the X3D world. The X3D root node called the <Scene> indicates the beginning of the scenegraph. The X3D scenegraph child nodes that constitute the elements of the X3D scene follow subsequently.

Each node contains fields that store the data associated with that node. Fields may contain a single value or an array of values. Field values may be integer, Boolean, single/double precision floating point and strings. In the next section, some basic graph concepts are mentioned.

2.2.3 Scenegraphs

Scenegraphs are a model centric approach to 3D authoring. As mentioned previously, a scenegraph is a data structure, wherein each node of the data structure contains some aspect of the 3D model or scene. The nodes may contain aspects of the 3D scene such as a description of the geometry of objects, relative locations of the objects, transformations, materials, etc. present in the 3D scene. The scenegraph structure provides a logical and often spatial ordering of the 3D content present in the scene. An example of a logical ordering would be the expression of the relationship of a car to its occupant such that the

occupant is a child node of the car in the scenegraph. In continuing with the previous example, a spatial ordering would then result in the occupant moving when the car moves. By expressing these logical and spatial relationships in a 3D scenegraph, an effect of changing one node (parent), can be propagated to all the nodes that have a logical relationship with it (child nodes). Scenegraphs may be implemented as an array wherein operations are performed in linear time. However, this data structure with linear operation time may be often inadequate for relatively larger virtual scenes/worlds. Consequently, it is quite common to find trees being used as a scenegraph data structure.

Scenegraphs have been used in several 3D authoring tools and Application Programming Interfaces (APIs). Most retained mode 3D APIs use scenegraphs. The first API to use scenegraphs was the Programmer's Hierarchical Interactive Graphics System (PHIGS). Open Inventor [21] and Java3D [22] both use scenegraphs. As mentioned previously, both VRML and X3D have scenegraph based architectures.

The primary objective of this dissertation is to provide a framework for mining 3D content using different information sources and levels. For example, if a 3D scene is part of a web-page, there may be hypertext that can be indexed and used for retrieval. If a 3D scene has a semantic description, the TREE-D-SEEK framework must be capable of retrieving content using this higher level description. In the subsequent sections, background and work relevant to retrieving content using textual keyword content, scenegraph content, shape content and 3D semantic data is discussed.

2.3 Approaches to retrieving textual content based on Information Retrieval (IR)

The field of IR has traditionally involved mining unstructured and semi-structured textual content. Information retrieval techniques for textual content are classified broadly into two categories as shown in [23]. Semantic retrieval techniques attempt to "mimic" human understanding of natural language text. Semantic retrieval techniques are often used along with statistical retrieval techniques. Statistical approaches involve breaking words into terms. Generally, terms are words that occur in a query or a corpus. A query is a "search-string" used to find a relevant match from the target corpus. Querying can be ad-hoc or routing based. Routing queries are typically topic filters, i.e. each term in the query "routes" the searching system to a predefined topic. Adhoc queries are typically arbitrary search strings. A corpus is a collection of files or documents.

Some search engines can also recognize phrases as terms. A phrase is a combination of words in a query or corpus. Some engines may also break documents into strings of n consecutive characters [24]. "n-grams" may be extracted by moving a window of n characters in length through a document or query one character at a time. Numeric weights are commonly assigned to both query and document terms. Weight of any term in the document is a measure of how well the term can identify uniquely the document. Weight of any term in the query is a measure of how important the term is in identifying the document in the corpus.

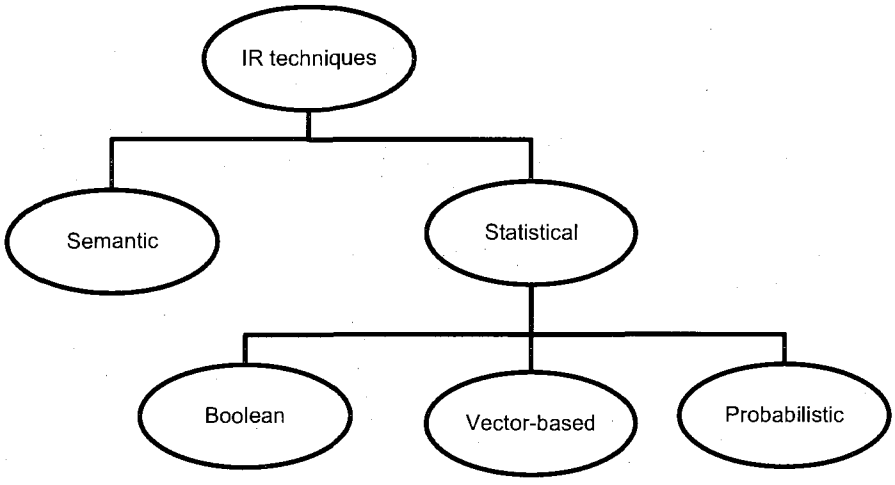


Figure 8: IR technique classification.

Statistical techniques can be further classified into Boolean, Extended Boolean vector space, and Probabilistic. In the Boolean approach, queries are formed by logical ANDing or logical Oring each query term. A document will match the query only if both terms that have been ANDed in the query have been found or if either of the terms appearing in the query is found if the terms had been ORed. This method is incapable of producing ranked output. In the extended Boolean approach, weights are assigned to the terms in the query and document. P-norm models are constructed where given a query consisting of n query terms $t_1,t_2,\ldots t_n$ with corresponding weights $w_{q1},w_{q2},\ldots w_{qn}$ and a document D with corresponding weights w_{d1}, $w_{d2},\ldots\ldots$ w_{dn}, similarity functions of the P-norm model are computed by (1) and (2).

$$SIM_{AND}(d, (T_1, W_{Q1}) \text{ AND } \ldots \text{AND} (T_N, W_{QN})) = 1 - \left[\frac{\sum_{i=1}^{n} (1 - w_{di})^p . w_{qi}^p}{\sum_{i=1}^{n} w_{qi}^p} \right]^{1/p} \quad \text{where } 1 \le p \le \infty \qquad (1)$$

$$SIM_{OR}(d, (T1, W_{Q1}) OR \ldots OR (T_N, W_{QN})) = \left[\sum_{i=1}^{n} \frac{w_{di}^p . w_{qi}^p}{\sum_{i=1}^{n} w_{qi}^p} \right]^{1/p} \quad \text{where } 1 \le p \le \infty \qquad (2)$$

At $p = \infty$, the extended Boolean model transforms into classic Boolean. At p=1, the extended Boolean becomes a vector space model. A probabilistic retrieval model is similar to the other statistical retrieval methods. According to [25], a distinguishing feature separating probabilistic methods from other statistical retrieval models is the use of formal probability theory and related statistics to provide estimates for relevance ranking. The vector space approach is discussed next.

2.3.1 Vector space approach

In the vector space approach, a document vector in multidimensional Euclidean space is used to represent an individual document. Each distinct term is a dimension in this space. Each term is assigned a numeric weight to indicate the ability of the term to act as a descriptor for the document. A given term may receive different weights in different documents. The weights assigned to terms in a document can represent the coordinates of the document vector in the Euclidean space. The corpus or collection of documents may be represented by using a document by term matrix where each row is a document and each column is a term and an entry at i^{th} row and j^{th} column indicates the weight of term j in document i.

The most common weighing scheme used to assign weights to terms in documents is the Term Frequency Inverse Document Frequency (TFIDF) weighing scheme. Term frequency (TF) is the frequency of occurrence of a given term in a document. Term frequency is a local document specific statistic. Inverse Document Frequency (IDF) maps the frequency of the occurrence of a term over the entire corpus. IDF is defined as $\ln(\frac{N}{n})$ where N is the total number of documents in the collection and n is the number of documents containing the term. IDF is zero if a particular term appears in the entire document. This indicates that the term may not be a good descriptor for the document or any document in the collection.

The mathematical product of TF and IDF basically indicates a strategy of identifying and assigning heavier weights to terms that occur frequently within a document and do not occur too frequently in other documents or terms that occur moderately within a document and over documents in the collection. The weights need to be normalized to account for variations in document size.

Once the query and the documents in the corpus have been weighted and assigned vectors in document space, a similarity measure is required to compare objectively the query and the documents. This numeric score may be a measure of similarity or dissimilarity. The most frequently used similarity measure used in vector space retrieval is the dot or cosine product given by

$$SIM(Query, Document) = \sum_{i=1}^{n} QueryTerm_i \bullet DocumentTerm_i \quad (3)$$

2.4 Approaches for retrieving content based on structure

The need for information exchange, interoperability, inferring and searching are motivating principles behind organizing data in virtually every domain. Each of the above mentioned principles clearly require the structure to be the least common denominator. Structure refers to the requirement of data to adhere to rigid schemata. The Extensible markup language (XML) has now become the lingua franca to represent structure. XML versioning is critical in several applications such as collaborative-authoring, warehousing, and software configuration versioning systems. XML versioning in web based crawling is also important for identifying the "permanence" of links to web pages.

For the purpose of this dissertation, scenegraphs that will be mined are XML based trees. Also, the TREE-D-SEEK framework is capable of retrieving content, based on semantic metadata marked up in XML. In the following sections, related work in software versioning, structure-matching and available XML document comparator algorithms is presented.

2.4.1 Software versioning systems

Difference detection for documents has been studied extensively. The GNU *diff* utility shipped along with UNIX installations since 1974 is an example of a popular file comparison utility for comparing two text files. The diff utility uses the Longest Common

Subsequence (LCS) [26]. Furthermore, several versioning systems make use of the diff utility to store deltas are the Revision Control System (RCS) [27] and the Source Code Control System (SCCS). A more recent example of a version system is the Concurrent Versioning System (CVS) [27] that is built on top of RCS.

Both RCS and SCCS are edit distance (delta) based. In RCS, the most recent version of a document is kept unchanged and all older versions of the document are stored as reverse delta. In SCSS, the original version of a document is stored along with forward deltas and timestamps. The above mentioned tools are not capable of supporting structure based queries as they process the documents as a sequence of text strings and are therefore not ideal for XML file versioning [28].

2.4.2 Structure matching algorithms

XML document structure is tree based; therefore, tree structure matching can be used in the process of structure matching of XML documents. Tree structure matching algorithms can be classified into ordered tree matching and unordered tree matching. Ordered trees are trees that have a well- defined sibling–sibling ordered relationship for every node in the tree. Unordered trees are trees that have no specific "order" in the relationship between children of a parent node. Ordered and unordered matching algorithms can also be classified into top-down and bottom-up algorithms based on how the trees are traversed and compared with each other.

Two trees are isomorphic if and only if there is a bijection between the vertex sets of the trees preserving the structure of the trees [29]. Clearly, in the case of ordered tree isomorphism, the bijection should provide a mapping between nodes in each of the trees preserving the structure of the root and the structure of the sibling nodes. In unordered tree isomorphism, the bijection should preserve the structure of the tree wherein there is a mapping from a vertex in one tree to a vertex in the other tree preserving the structure of the root, and parent child relationship.

The subtree isomorphism problem is a generalization of the tree isomorphism problem. It involves identifying whether a given tree is isomorphic to a subtree of another tree. A generalization of the subtree isomorphism problem is the maximum common subtree isomorphism problem wherein the objective is to determine the largest common subtree between two trees [29].

2.4.3 Related work in tree matching

In [30], a bottom-up algorithm with O(n) for identifying isomorphism between rooted unordered trees with n nodes is described. The algorithm assigns integers to the nodes of the two trees in a bottom up fashion. The two trees are identified as isomorphic if and only if the roots of the two trees have the same integer value. A tree to tree correction algorithm based on edit operations was proposed in [31]. In [32], a restriction of [31] is presented, wherein a top down strategy is used and only nodes on the same level are matched. The time complexity of this algorithm is $O(n_1n_2)$ time, where n_1 is the number of nodes in $Tree_1$ and n_2 is the number of nodes in T_2. In [33], an edit distance based

algorithm based on a post-order, dynamic programming approach is proposed. In [34] and [35], bottom up subtree isomorphism algorithms are presented and the run time complexity is of the order $O(n_1+n_2)$ where n_1 and n_2 are the number of nodes in each of any two trees T_1 and T_2 respectively. The strategy used in order to achieve this runtime complexity is a bottom up approach wherein the maximum common subgraph is identified by computing a directed acyclic graph that partitions the two trees T_1 and T_2 into isomorphism equivalence classes.

2.4.4 XML comparator algorithms

The Xydiff algorithm proposed in [36] is used to compare two XML documents using a bottom up approach. Each node is traversed using a bottom up approach, and each node is assigned a signature and a weight. The node signature is a bottom up hash value of the current node content as well as its children. The weight is a bottom up value proportional to the size of its subtree. Subsequently, in a top-down traversal, nodal signatures are compared. If the nodal signatures are not identical, then the children are inserted into a priority queue based on the weight. The heavier weighted subtrees are compared first. If there is an identical match between two subtrees, then the node signature and weight is pushed upwards to the parent. The weight of the subtree dictates the level to which this value is propagated. The Xydiff algorithm fails when the leaves of the trees are changed. The Xydiff algorithm also uses XML specific attributes such as ID and other heuristics to obtain O (n log n) efficiency.

In X-Diff [37], unordered XML documents can be compared. It uses a more exhaustive edit based approach wherein each node is assigned a hash value in a top-down traversal. If root nodes have the same signature, the trees are identical. If the roots do not have the same signature then minimum edit cost algorithms is used to compare trees. The X-diff execution time is of the order of $O(n^2 \cdot d \cdot \log d)$, where d is the total degree of any node in the tree.

In XMLTreeDiff [38], DOM-Hash [39] is initially used to reduce the size of the two XML documents by removing the maximum common subtree. Then it uses another minimum edit tree algorithm to compare the two simplified trees.

2.5 Content based 3D retrieval

In comparison to retrieval of other types of multimedia, 3D content retrieval is relatively recent. Owing to improvements in 3D acquisition, production and consumption capabilities, growth of 3D applications for visualization, training, entertainment, etc. are certainties. As in other types of multimedia, content based methods for retrieving 3D content is an important research area. In content based 3D searching, the goal is to identify an objective measure of similarity to search, compare and retrieve a user query to the target corpus. The query provided by the user contains 3D content that may be used to find and retrieve similar content from the target corpus. A generic strategy in content based searching is that 3D objects are modeled as objects in a vector space, and a distance function is used to measure the similarity or dissimilarity between objects. The similarity between objects may be based on the global geometric similarity of the two 3D

objects. Typically, the global geometric similarity may be calculated using either a *direct geometric matching* strategy wherein two 3D objects are directly matched based on transformation cost associated in converting one object to the other, or a *descriptor based* strategy wherein features extracted from the 3D objects are used for indexing, comparison and retrieval. Direct geometric matching strategies is therefore not scalable for large 3D corpuses as this would involve evaluating each pair of models for every model in the corpus once the query model has been submitted. A descriptor based approach is relatively more scalable as features can be extracted apriori and offline for the corpus and compared with the features of the query model at the time the query model has been submitted. Shape descriptors need to be discriminating, run-time efficient and compact to store. 3D shape matching methods can be classified as shown in Figure 9. The classification categories are not mutually exclusive. In other words, a skeleton matching algorithm may also be classified as a global feature distribution. [40].

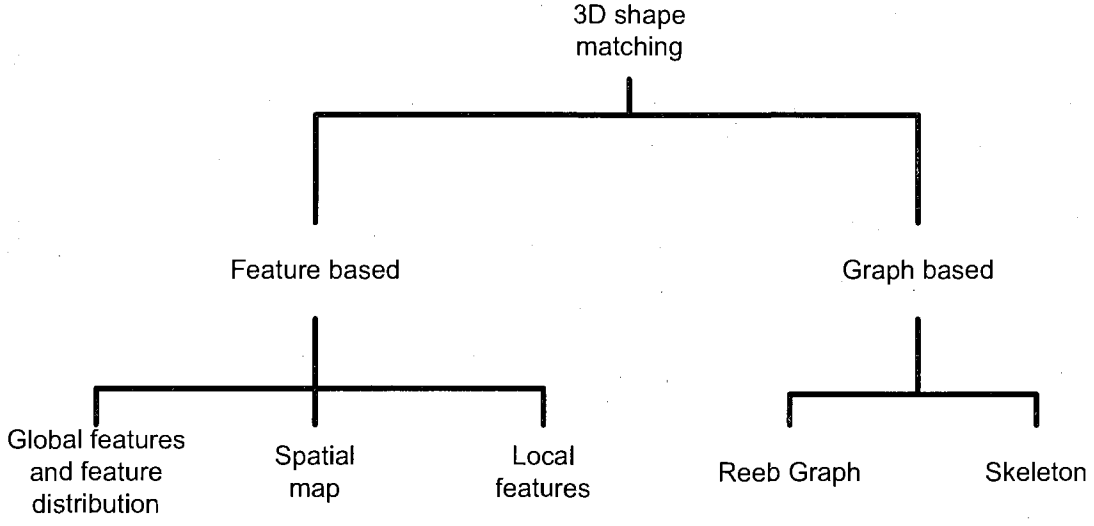


Figure 9: Classification of shape matching algorithm classification [40].

3D shape matching methods may be broadly classified into feature based or graph based methods. Global features, global feature distribution, and spatial maps features can be represented as a single vector in Euclidean space of n dimensions where n is fixed apriori for a 3D corpus [15]. Some related work in global feature and global feature distribution based algorithms is described subsequently.

2.5.1 Related work performed in global feature, global feature distribution and spatial map based techniques

In [41], calculating global features such as moments, Fourier transform coefficients, and volumes from 3D meshes is described. In [42], bounding boxes, wavelet based descriptors for 2D and 3D objects are described. The descriptors are defined using MPEG-7. An example of a global feature distribution based technique is the D2 algorithm [43]. The D2 shape distribution provides the distributed of Euclidean distance between randomly selected two points on the surface of an object. It is a rotation invariant descriptor.

In spatial map based techniques, descriptors capture the spatial location of objects. Generally, pose normalization is required. An example of spatial map technique is the shape histogram technique found in [44] . Three types of descriptors fall under this category namely shell descriptor, sector descriptor and shell & sector descriptor corresponding to how the 3D space is partitioned. In the shell model, the 3D space is partitioned into concentric shells. In the sector model, the 3D is partitioned into sectors emerging from the center. In the spider web model, the shell and sector model are

combined to partition the 3D space. The shell descriptor represents the distribution of distances of surface points from the center of mass. It is therefore a 1D descriptor. The sector descriptor represents the distribution of surface points as a function of spherical angles. Spherical harmonics based rotational invariant features are suggested in [45]. By decomposing a 3D model into functions on concentric spheres and using spherical harmonics to discard orientation information, the shape descriptors are made orientation invariant. 3D Zernike descriptors from voxelized models are presented in [46].

2.5.2 Related work in local feature based similarity.

In local featured based approaches, surface shape is evaluated around localized points on the boundary of the shape. An example of this technique can be found in [47]. In this paper [47], two 3D shapes are compared based on their curvature distributions generated from their deformed meshes. Local feature based similarity techniques are difficult to compute.

2.5.3 Related work in graph based methods

In graph based methods, a graph of the inter-connectivity structure of shape components constituting a shape is extracted from the 3D object and matched. Graph matching based on edit distance is NP hard. Graph matching using maximum common subgraph techniques is NP complete.

2.6 Retrieval based on standards

The benefits of having standards for multimedia representation such as the Joint Photographic Experts Group (JPEG) standards or Moving Pictures Experts Group

(MPEG) standards are apparent. Clearly, standards allow addressing interoperability issues, thereby increasing the number of consumers for the products and finally resulting in lowering of production costs. Also, standards enable the creation of suites of tools to facilitate the acquisition, development and deployment of products. Support for 3D content specification may be found in MPEG-7 and MPEG 21.

The MPEG-1 was developed in 1992 and the goal was to encode moving pictures and sound to fit into a CD-ROM. MPEG-1 uses the Standard Interchange Format (SIF) which stipulates a 352x240 NTSC at 1.5 Mbps [48]. MPEG-2 was established in 1994 and is capable of broadcast quality video at its specified bit rates between 3-10 Mbps. The MPEG-3 standard was not popular and was dropped. The original target market for MPEG-3 was High Definition Television (HDTV). The MPEG-4 was finalized at the end of 1998 and the target market is the television and the World Wide Web (WWW). MPEG-4 allows encoding at rates varying from 2Kbps to 5Mbps. The MPEG-4 provides the capability to integrate content of synthetic 3D content and is compatible with the VRML standard mentioned earlier. More detailed description of MPEG-7 and MPEG-21 are provided below.

2.6. 1 MPEG-7

MPEG-7 was proposed in July 1996 and the goal of MPEG-7 standard is to address the necessity for improved search and retrieval of multimedia content. Multimedia content refers to still pictures, video, audio, 3D models and graphics. MPEG-7 is solely a content description capability or to be the "the bits about the bit" [49]. It was not designed to

replace any of the existing MPEG standards. It provides capabilities to define and describe already captured and stored (offline) content and streaming or broadcasting environments (real-time). MPEG-7 is an isolated description file that may be available with the actual multimedia representation format file. MPEG-7 is capable of describing scenes using object based composability. A noteworthy feature of MPEG-7 is that it supports a unified framework to support both low-level and high-level feature description. MPEG-7 toolkit supports the following functionalities [49]:

- Descriptors.
- Description Schemes.
- Description Definition Language (DDL).
- System tools.

A Descriptor represents a content feature. It specifies the syntax used to specify a feature. A description scheme specifies the syntax and semantics that descriptors representing the multimedia content would adhere to. A Description Definition Language is the language used for creating and defining the descriptors and creating or extending the associated scheme. MPEG-7 DDL is XML based. The Description Schema of MPEG-7 is W3C XML Schema based. System tools refer to tools that support creation and management of intellectual property, storage of descriptions etc. [49].

2.6.2 MPEG-21

The primary objective of the MPEG-21 standard is to support the delivery and consumption of multimedia over an extensive range of devices and networks and also to provide mechanisms to define and distribute transparently the digital rights/

permissions/intellectual property rights associated with the multimedia content. MPEG-21 is also known as a multimedia framework, open for specifying any type of media, content representation, digital rights/intellectual property rights, delivery mechanism, and the use of content. Indeed, the MPEG-21 framework provides support for content producers, distributers and consumers involved with any media and any device. The MPEG-21 framework defines two main concepts namely a *Digital Item* and a *User.*

A Digital Item is the basic multimedia unit available for consumption. The Digital Item concept recognizes the fact that it is not necessary for a multimedia application to be solitary. A Digital Item may be associated with another Digital Item within a multimedia application. For instance, a video file may be associated with a still picture (video cover page) and a transcription file. The digital items can be identified by using the MPEG-21 Digital Item Identification (DII) standard. The structure of individual digital Items relating to each other within a multimedia package can be defined using the MPEG-21 Digital Item Declaration (DID) standard.

The *User* in MPEG-21 framework is an entity that interacts with the Digital Item. The MPEG-21 Digital Item Adaptation (DIA) standard and tools can be used to describe the network conditions, bit stream representation and other usage based environment parameters. The MPEG-21 defines both producers and consumers as Users. A consumer and a producer are both users but with different rights based on interactions with other users.

2.7 Retrieval based on ontologies and semantic metadata

Ontologies and ontology based retrieval is an area of active research interest. The goal is to improve "machine understandability and reasoning" to assist in satisfying the "need" behind the search query as opposed to a purely statistical search or a search based on pure meta-data only. According to Gruber [50], "ontology is an explicit specification of a conceptualization." A concept is an abstract, simplified view of the world. A concept is generally language independent and may be interrelated to other concepts. The Semantic web, which has attracted a lot of attention, has been defined as the "conceptual structuring of the web in an explicit machine readable way" [51].

Syntactic metadata is metadata that does not provide any contextual or domain specific information but provides basic information such as the date of creation of a document, or author's name etc. Semantic metadata refers to metadata that provides contextual information. Semantic metadata may be defined using a domain specific metadata model or as an ontology instance. Ontology may be created using an ontology language. The proposed language stack of the Semantic web is shown in Figure 10.

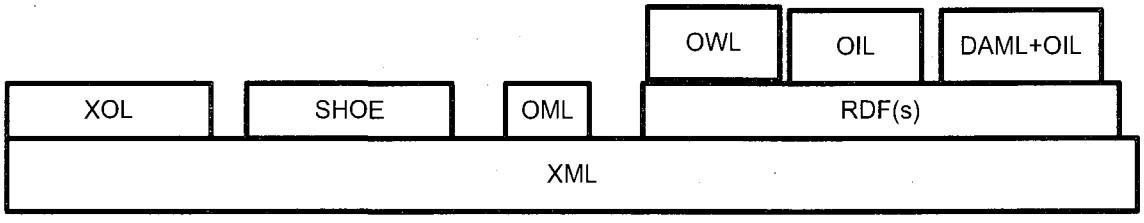


Figure 10: Semantic web language stack.

The language stack of the Semantic web can support several languages such as Ontology XML Language (XOL) [52], Simple HTML Ontology Extension(SHOE) [53], Ontology Markup Language (OML) [54], Resource Description Framework (RDF) [55] ,RDF Schema [56], Ontology Inference Layer (OIL) [57] , (DARPA Markup Language) DAML+OIL [58] and Web Ontology Language [59]. As seen in Figure 6, the underlying structure of the Semantic web is based on XML. Related work done in ontology versioning systems and 3D retrieval using semantic metadata and ontologies is described subsequently.

2.7.1 Ontology versioning systems

Ontology management and versioning systems are an area of active research interest. As the number of ontologies increase, there will be a need for managing ontologies. Several ontology management and versioning systems have been proposed. An example of an ontology versioning system is PromptDiff [60]. PromptDiff can perform structural comparisons using sub-graph isomorphism. AnchorPrompt [61] is a graph based tool for finding related concepts in different ontologies provided an initial mapping exists between the ontologies being compared. In [62], several important distinctions between versioning systems and ontology versioning systems were made. For instance, in [62], a distinction is made between versioning changes and inter-conceptual changes and a distinction is made between conceptual changes and specification changes. An example of an ontology mapping system is GLUE [63].

2.7.2 3D content and standards

The XMT-A standard is an extension of MPEG-4 for specifying audio-visual content using XML. It contains a subset of X3D nodes in the specification. In [64], semantic metadata for 2D/3D scenes is described using MPEG 7. A semantic graph representing the 2D or 3D scene was defined using MPEG-7. A user interface was described that enables selection of a high level description of an object resulting in the highlighting of the associated low level geometry of the object in a VRML browser. Also, the capability to select the low level geometry of the object in the browser resulting in the viewing of the semantic relationship of the selected object is possible. The content description file using MPEG-7 is a stand-alone file and not part of the content file itself.

In [65], the use of domain specific annotations and Ontoworld [66] to annotate existing virtual worlds is described. A strategy to index 3D scenes using MPEG-7 is discussed in [67] wherein an annotation model is described for 3D models. The annotation model used MPEG-7 extended with 3D specific locators.

2.7.3 3D content and ontologies

In [68], a combination of X3D and RDF technologies is used to provide semantics to 3D virtual environments. Scene independent, domain specific, reusable ontologies in RDF provide the high level conceptual relationship definitions. These definitions are used in providing semantic annotation directly within an X3D file. As a result, semantic information using this strategy involves at a minimum two files.

An interesting approach to 3D content retrieval based on shapes and ontologies is discussed in [69]. In this work, a knowledge based framework for the annotation of 3D shapes is discussed. The shape annotation is done by deriving the possible functionality of the 3D object. Geometric recognition tools are used to derive functionality of the 3D objects based on their shapes.

2.8 Discussion and summary

As mentioned in Chapter I, there is a need for a unified retrieval strategy wherein a 3D system is capable of retrieving 3D content based on indexing information sources at different information levels. Existing retrieval solutions use closed systems to provide a generic 3D retrieval capability and a fixed set of indexing and matching techniques for retrieving content. An open, extensible, framework is required for 3D search engine developers so that customized solutions for retrieving 3D content may be implemented.

In this chapter, 3D model representations were studied. 3D models representations impact the choice of indexing and matching algorithm. For example, if a corpus consists of 3D models wherein each 3D model is represented using polygonal meshes; retrieval using scenegraph structure will suffer from poor precision. Related work in indexing, matching and retrieving content using text-based, structure-based, shape based and semantic metadata is also discussed in this chapter. This study of existing indexing and matching algorithms provides an understanding of the process flow for retrieving content based on indexing related information sources and information levels.

In this research, a unified strategy and framework for retrieving 3D scenes is proposed. The strategy is to retrieve 3D content by indexing and matching related information sources at different information levels. A framework is proposed that implements the retrieval strategy. The retrieval strategy is discussed next.

Chapter III

TREE-D-SEEK RETRIEVAL STRATEGY

A unified 3D retrieval strategy and framework that supports retrieval of 3D content by indexing information sources at different information levels provides several benefits. This retrieval strategy allows a user to query a 3D corpus based on the user's individual level of ease and skill. For instance, a non 3D user can use metadata to query the corpora. A 3D modeler can create a 3D model and use the model as a query to retrieve similar models. A unified, open framework provides the 3D search engine developer to create a customized search solution for specific corpora and requirements. A unified framework supports the creation of a test bed to evaluate indexing and matching algorithm for different information sources at different information levels.

In this chapter, a unified strategy for retrieving 3D models is discussed. The strategy is to provide a unified approach for retrieving 3D scenes wherein information sources are indexed and matched at the syntactic-metadata, shape, scenegraph and semantic metadata levels. The processes for indexing different information sources may vary. In order to build a unified software framework that can support retrieval of 3D content based on indexing different information sources, the individual indexing processes for each type of information source need to be reviewed and any commonality in the retrieval process flow must be isolated. A brief discussion of the process flows for retrieving content based on indexing the different information sources is presented in this chapter. Next, a

common process flow that can be used for indexing and matching different information sources is derived. This common process flow may be then used in designing a software retrieval framework and is presented in the next chapter. The retrieval strategy is discussed next.

3.1 3D scene retrieval strategy

3D scenes are not necessarily isolated islands, available only as self contained units. For instance, web based 3D scenes may be related to the embedding web page. Hypertext, keywords, file names and associated Uniform Resource Identifiers (URIs) can provide information about 3D scenes. 3D scenes may be scene-graph based. A scenegraph may provide further information about the composition of the 3D scene. A scenegraph may have a hierarchical structure that can be used for structure matching. A 3D scene may contain semantic annotations based on domain specific ontologies. Therefore, to improve retrieval relevance and allow for different querying mechanisms, a 3D search engine framework can support retrieval based on:

- Mining external textual content associated with the 3D scene.
- Mining scenegraph content and structure.
- Mining low level 3D content.
- Mining 3D-scene semantic annotations.

Based on the retrieving processes reviewed in Chapter II, a strategy for retrieving 3D content is shown in Figure 11. The components shown in Figure 11 are derived from components that are common to a generic IR retrieval process. *Corpora* may be a

collection of 3D scenes or other content such as web pages, metadata, semantic annotations relevant to the 3D scenes present in the corpora. *Crawling* may be required to discover 3D and related content from any corpora and transferring the content to a predestined location such as in a local file system. *Indexing* is a process of creating a data structure for faster retrieval. In 3D content and multimedia retrieval, indexing involves identifying and extracting features from the content, creating descriptors for the extracted features and storing the descriptors. A feature is a characteristic of the data selected to compare and evaluate the data present in a corpus. For instance, shape of 3D objects may be a low level feature used for retrieving 3D objects. A descriptor is a representation of a feature that contains a numeric value or numeric values allowing objective evaluation of the corresponding feature. An example of a 3D descriptor can be a histogram of distances between random surface points on the 3D object. In the proposed retrieval strategy, features corresponding to text, low-level, scenegraph and semantic annotations are identified and corresponding descriptors are created and stored.

The *searching* process involves formulating a query, handling the query, matching the query and returning the results obtained from the search. Formulating a query requires identifying the type of query and providing the selected query to the retrieval system. This strategy supports querying by text, scenegraph content, semantic annotations and 3D shape. The choice of query may be dependent on the matching algorithm selected by the user.

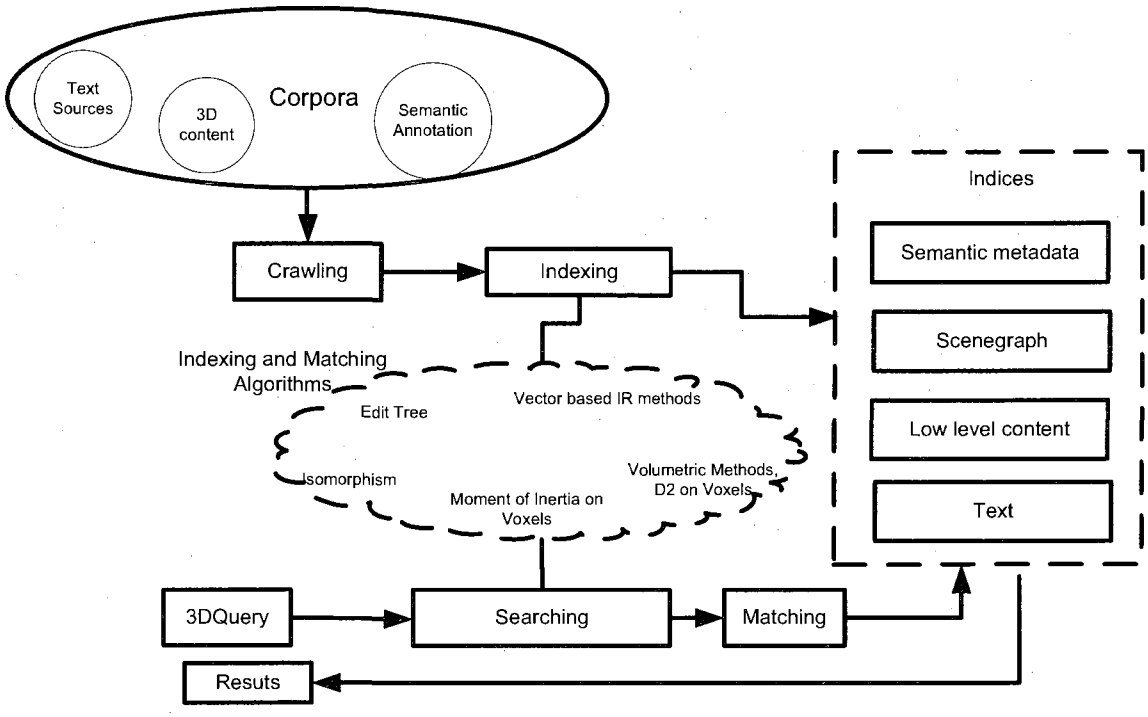


Figure 11: TREE-D-SEEK retrieval strategy.

Matching refers to the process of using a similarity (or dissimilarity) measure to compare the query to the indexed 3D content. For instance, Euclidean distance may be used as a similarity measure to compare a pair of descriptors. Based on the score obtained from using the similarity measure, top hits or results of the search are returned back to the user based on the similarity measure. Descriptors are generated based on the same algorithm that was used in the indexing process. The indexing process is done apriori. Descriptors are generated for each document in the corpora. The matching and retrieving processes are initiated when the query is submitted to the retrieval system.

As mentioned previously, the components shown in Figure 11 correspond to generic process flow in an IR retrieval strategy. The proposed retrieval strategy is unique in the concept of indexing corpora for 3D content retrieval at the syntactic metadata, low-level content, scenegraph structure and semantic annotation level.

3.2 Text retrieval

Although textual querying for 3D content may yield significant noise, textual querying remains the most frequently used query interface for any multimedia retrieval system [70]. An example of textual indexing and searching is the Google image search [71]. Examples of text sources in the retrieval of 3D content may include file names, URIs, Hypertext Markup Language (HTML) web pages, etc.

The generic process flow for text indexing is shown in Figure 12. This also represents the baseline process flow in the TREE-D-SEEK framework for text indexing.

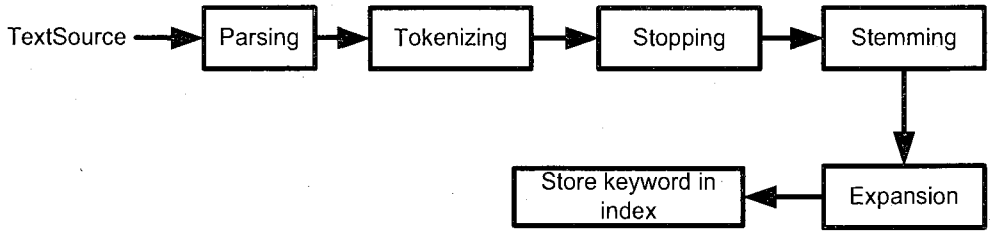


Figure 12: Text indexing process flow.

Since the text source may store information based on its own particular schema or encoding, a parser may be required to extract the relevant features. The features extracted

may contain a stream of text. A word tokenizer may be required to isolate words from the stream. The tokenized words are then cleaned by removing stop words, i.e. words such as "and" that are not sufficiently discriminating. The stemmer may be used to remove inflected words. For example the word 'dogs' after stemming becomes 'dog'. Once stopping and stemming have been performed, keywords may be expanded by performing a lookup on a thesaurus. Once, this analysis of the token stream is performed, the keywords are indexed and stored.

The searching process flow for retrieving based on keywords is shown in Figure 13. First the query is indexed and then the keyword descriptors are scored using a matching function to score the query with the stored keywords and the top results of the search or top hits are returned back to the user.

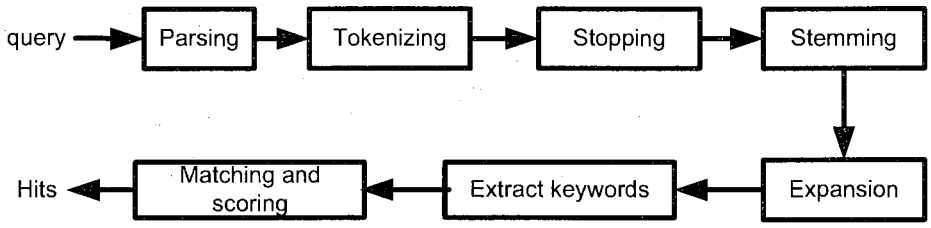


Figure 13: Searching process flow.

3.3 Scenegraph based retrieval

A scenegraph is a data structure, wherein each node of the data structure contains some aspect of the 3D model or scene. The nodes may contain aspects of the 3D scene such as

the geometry of objects, relative locations of the objects, transformations, materials, and etc. that are present in the 3D scene. The scenegraph may contain additional metadata related to the contents of the 3D scene. To our knowledge, no previous work has focused on retrieving 3D content based on matching scenegraph content and structure.

For the purposes of this research, a scenegraph is a rooted, acyclic tree. The nodes in the tree provide a ordering of the 3D objects present in the scene. Consequently, a scenegraph may be mined based on both the metadata content present in the scenegraph and also based on the structure of the scenegraph itself. The generic process flow for indexing 3D scenegraphs is presented in Figure 14.

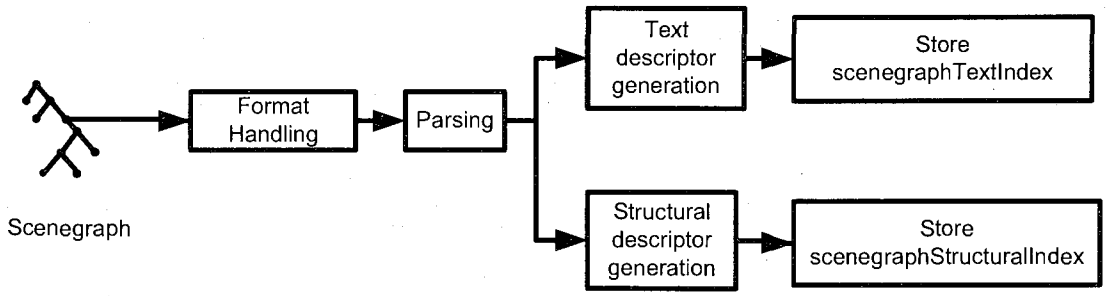


Figure 14: Scenegraph indexing.

3D scenes may be authored using a variety of tools and languages and encoded into a variety of file formats. Also, not all 3D scenes may have an underlying scenegraph. For instance scenes that have been authored using OpenGL may not have an underlying scenegraph. The problem of creating a common scenegraph for any 3D authoring language or technology is addressed in [72]. In this research, a rapid prototyping

framework is used to extract scenegraph content and structure from a source and to store the same in a common scene format [72]. Next, certain nodes of the scenegraph may be used for feature extraction. For example, the metadata node in X3D may be specifically targeted and its contents examined. The feature extractor can select specific nodes and create the respective indices for storage.

Once the scenegraph content and structure have been extracted, descriptors for both structure and content are created. The process flow for searching based on scenegraph structure and content is shown in Figure 15. The process begins by first indexing the query without storing the descriptors of the query in the index. Next, a similarity (or dissimilarity) metric is used to score the descriptor with the available descriptors in the index. The top scored matches are returned.

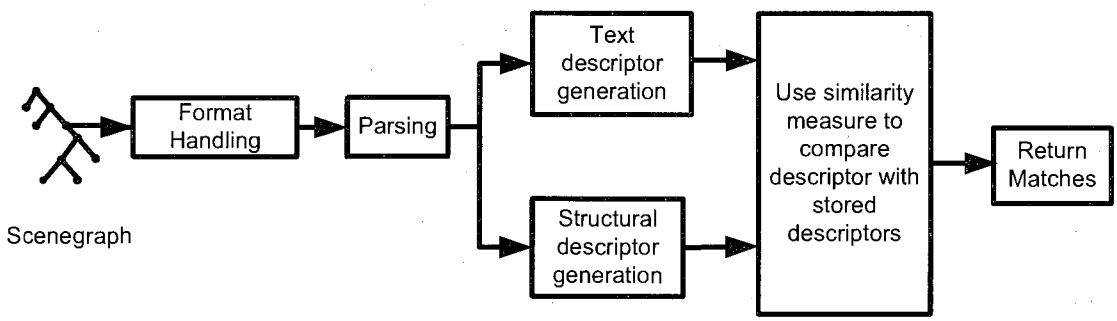


Figure 15: Scenegraph based searching process flow.

3.4 Shape retrieval

Shape based retrieval algorithms in the TREE-D-SEEK framework can only be used for retrieving isolated 3D models. The most widely used approach for comparing the

similarity between two 3D models is to evaluate the global geometric similarity between them. For the purposes of this research, local similarity based approaches are not considered. A common approach is to use descriptors to represent the global geometry of 3D models. 3D descriptors can be feature vectors wherein a vector of numeric values is used to represent the shape of the 3D model. 3D descriptors may be statistical in nature such as descriptors containing histogram summarizations of the shape of the object. 3D descriptors may be graph-based descriptors wherein characteristics of the topology of the 3D model may be summarized. The process involved in creating these descriptors is shown in Figure 16.

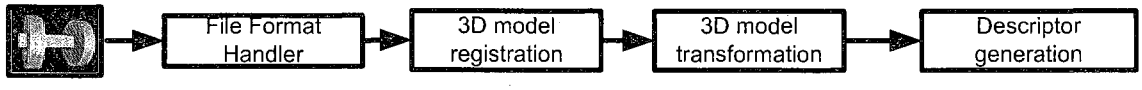


Figure 16: 3D shape indexing.

Since 3D scenes in the corpora may be available in different 3D formats, a file format handler is required to translate the scene into a 3D format that can be used for extracting features. Since 3D models are considered similar under translation, scale and rotation operations, a preprocessing step is required to register the 3D model into a canonical coordinate system. In the next step, the model is transformed into a 3D representation from which the desired features can be extracted. For example, a 3D mesh model may be transformed into a voxelized model. The voxelized model may be transformed using spherical harmonics to obtain numeric descriptors from the 3D model.

The searching process also uses the process flow shown in Figure 16 to generate the descriptors for the query. Then, a similarity measure is used for scoring the match between the query descriptors and the stored scene descriptors. The top hits are returned back.

3.5 Semantic annotations based retrieval

The vision of the Semantic web and the necessity for improved recall and precision in 3D retrieval techniques have resulted in significant interest in providing interpretable markup of 3D objects and scenes. If available, 3D scene annotations can provide several advantages.

Semantic annotations may improve the relevance of retrieved content. Retrieval of 3D scenes based on free textual annotations may be error prone as the annotations are subjective. Low-level content based retrieval may be comparatively more reliable as the features used in retrieval are objective and automatically extracted [73]. However, content-based retrieval methods for 3D scenes are based only on geometric matching of 3D models and do not account for the semantics associated with the 3D scene.

3D scene annotations allow for richer queries that can target the semantic properties of objects present in 3D scenes. Improved *recall* and *precision* of 3D scene retrieval will result in efficient reuse of 3D content thereby decreasing 3D authoring time. Semantic annotations can improve understanding, user interaction and navigation in 3D scenes.

Domain based semantic annotations can assist in the creation and development of 3D scenes [74].

Several works have focused on providing semantic annotations for 3D scenes. In [12], an unsupervised segmentation algorithm is used to partition 3D models into its basic components using a domain specific ontology. In [75], virtual reality scenes are integrated with semantic annotation for efficient querying and visualization.

In the proposed retrieval strategy, 3D scenes may be retrieved based on available semantic annotations. These semantic annotations of a 3D scene are assumed to be stored separate from the 3D scene. Also, the semantics of the annotations must be based on a domain specific ontology. Matching semantic annotations based on ontologies is a difficult problem.

A significant problem is that the underlying domain ontologies may differ in the conceptualization of a domain. As a result, the semantic annotations corresponding to each conceptualization may be related, but it is difficult to identify a relationship between the underlying ontologies. The annotations may also have been created using different ontology languages and therefore may have different structure and syntax. Generally, the current approach to comparing ontologies is to compare the linguistic and structural similarities of the underlying ontologies [76]. In linguistic matching, ontologies are matched using any textual label, class name, property name, individual name, etc. In structural matching, the strategy is to represent ontologies using graph formalism and to

perform graph matching for comparing ontologies. The generic process involved in indexing semantic annotations is shown in Figure 17. A format handler is required to parse the particular ontology language used to specify the annotations. For example, an OWL parser may be required to parse annotations formalized using the OWL language.

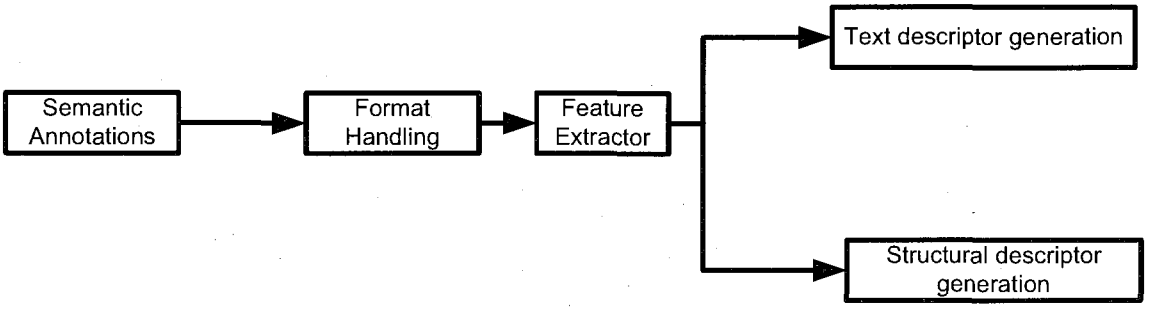


Figure 17: Semantic annotation indexing.

A feature extractor is required to extract relevant features that can be used to create structural and text descriptor. The descriptors can then be stored in the index.

3.6 Common process model

Based on the above mentioned processes involved in the indexing of 3D scenes, a common process model can be conceptualized. The common indexing process model encapsulates the processes into three containers as shown in Figure 18. The software architecture of the TREE-D-SEEK framework provides abstractions that support this process model. The indexing process is performed offline. As a result, only the descriptors for the query need to be generated at the time the query is submitted to the search system. The query descriptors are then matched and scored using a similarity (or dissimilarity) metric. The search process flow may be generalized into two phases. In the

first phase, the query is indexed without storing the descriptors in the index. The next phase involves matching and scoring the query descriptor with the stored descriptors in the index.

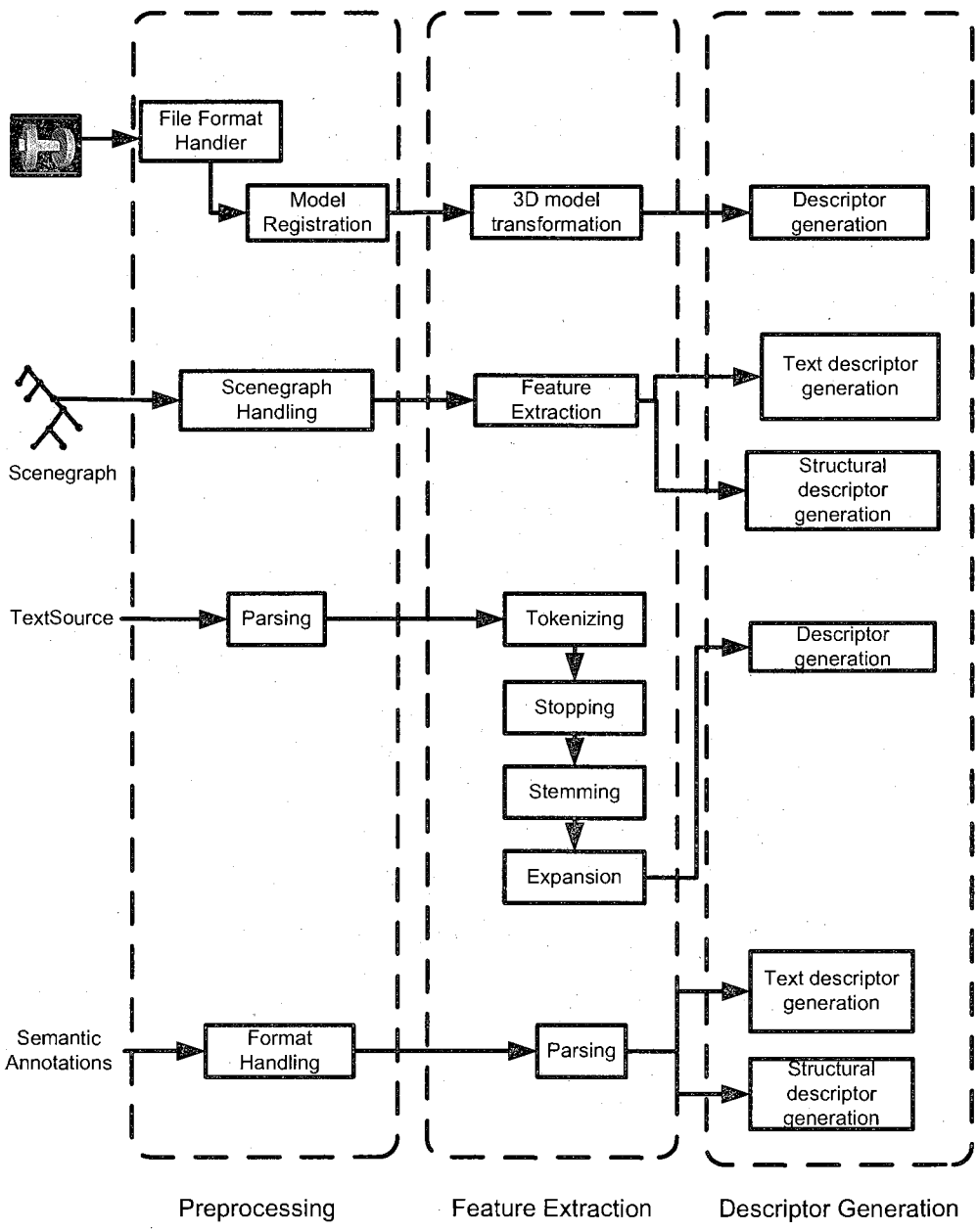


Figure 18: Generalized indexing process model.

3.7 Discussion

An open framework that supports retrieval of 3D scenes based on the retrieval strategy discussed in this chapter has not been proposed. A software architecture framework that is capable of supporting the common indexing and search process flows derived in this chapter has not been discussed or implemented. The TREE-D-SEEK framework implements the retrieval strategy discussed in this chapter. Several indexing and matching algorithms have been implemented as part of the TREE-D-SEEK framework as proof of concept that the framework can support retrieval of relevant information sources. The TREE-D-SEEK framework is discussed next.

Chapter IV

TREE-D-SEEK FRAMEWORK

The TREE-D-SEEK framework implements the retrieval strategy discussed in Chapter III. The proposed strategy is to retrieve 3D content based on text sources, scenegraph content, scenegraph structure and semantic annotations. The software architecture of the TREE-D-SEEK framework provides a unified interface to encapsulate the retrieval process for each of the above mentioned information sources. The rest of the chapter is organized as follows. First, the software architecture of the TREE-D-SEEK framework is discussed. Second, the software architecture is discussed from component, dataflow and interface perspectives. Third, a description of the implemented indexing and matching techniques implemented in the TREE-D-SEEK framework is presented.

4.1 TREE-D-SEEK: A component view

A retrieval process may be partitioned into two phases-the indexing phase and the querying phase. In the indexing phase, feature descriptors are created for the content and indexed. The process is performed offline and apriori to the querying phase. In the querying phase, descriptors are created for a query using the same indexing process. Next, the query descriptors are matched with the stored descriptors in the index and results from the search are returned in a predefined order to the user. Before the retrieval process can begin, the relevant content from available corpora must be discovered and made accessible for indexing.

The top level components in the TREE-D-SEEK framework are shown in Figure 19. For the purpose of this research, it is assumed that the relevant information sources from the corpora already reside on a file system that is accessible for indexing. However, the framework provides interfaces that support the integration of crawler implementations. The components have a one to one correspondence with class names in the TREE-D-SEEK framework. A brief description of each component is provided next.

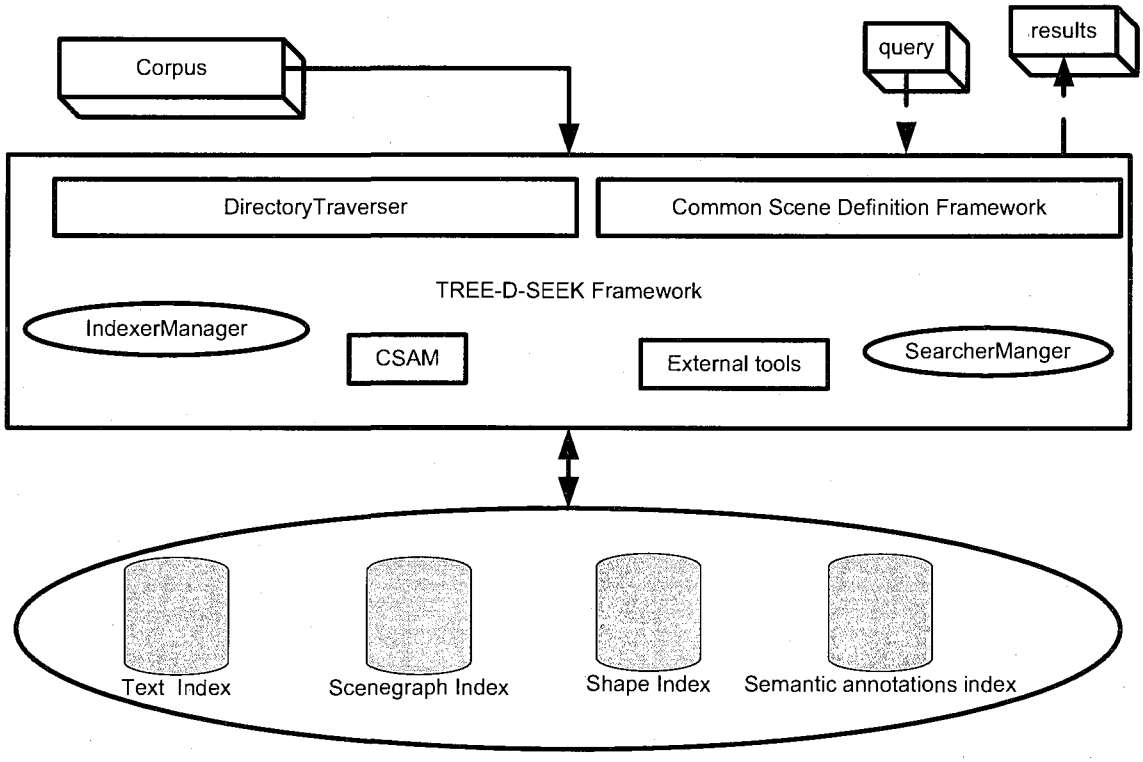


Figure 19: TREE-D-SEEK: A component view.

4.1.1 Common Scene Definition Framework

The Common Scene Definition Framework (CSDF) [72] is a 3D rapid prototyping framework. The CSDF is designed to be a superset of existing 3D technology. The CSDF framework itself is extensible and capable of transforming 3D scenes in different formats into a common scene definition. Synthesis modules in CSDF can synthesize or export from the common scene definition to the desired 3D representation of the user. The common scene definition elements have a mapping with the elements in the X3D standard. A major benefit of using the CSDF in the TREE-D-SEEK framework is the extensibility it provides in terms of parsing different 3D formats. Secondly, it provides a common scene definition for different 3D scenes.

4.1.2 DirectoryTraverser

The DirectoryTraverser is capable of recursively traversing a local file system. It is capable of discovering relevant content that can be fetched for indexing. The DirectoryTraverser identifies relevant content based on individual file extensions. For instance, a file that contains a ".owl" extension may be fetched for semantic annotation indexing. The interface and class description is shown in Figure 20.

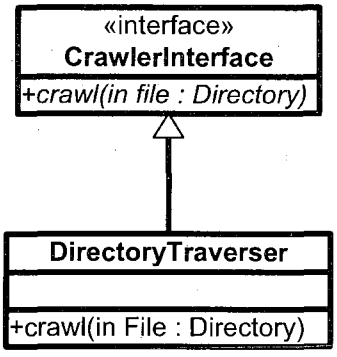


Figure 20: DirectoryTraverser.

4.1.3 IndexerManager

The IndexerManager component is responsible for handling and managing the indexing process in the TREE-D-SEEK framework. The IndexManager delegates the indexing of content to the respective content indexer. For instance, shape based content is sent to the shapeIndexer component. The IndexerManager class is shown in Figure 21.

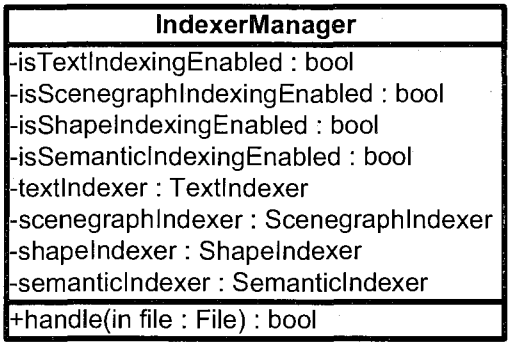


Figure 21: IndexerManager.

4.1.4 SearcherManager

The responsibility of the SearcherManager component is similar to the functionality of its counterpart- the IndexerManager component. The SearcherManager class is responsible for handling the searching process in the TREE-D-SEEK framework. The

SearcherManager delegates the handling of the query to the respective content searcher. For instance, shape based content is handled by the shape searcher. The SearcherManager class is shown in Figure 22.

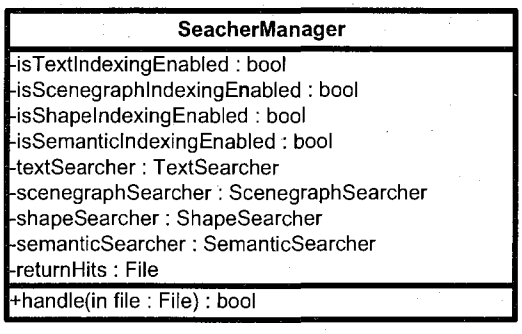


Figure 22: SearcherManager.

4.1.5 Matcher

The matcher component is responsible for scoring the query with the indices. The matcher component scores the query based on a predefined similarity measure and returns the matches in decreasing order of relevance.

4.1.6 Common Scene Annotation Modeler (CSAM)

The granularity of the retrieved content using the TREE-D-SEEK framework is at the 3D scene level. In other words, entire 3D scenes are returned as relevant results for each type of query. The primary objective of the Common Scene Annotation Modeler (CSAM) is designed to provide an intra scene querying and retrieving capability. For instance, the CSAM must be capable of retrieving individual 3D objects present in a particular 3D scene. To provide this capability in the TREE-D-SEEK framework, the strategy is to transform the common scene definition generated by the CSDF framework into a formal specification based on the TREE-D-SEEK semantic annotation model. The process must

also allow for manual human annotation. At the time of writing this dissertation, the CSAM module is not capable of fully specifying a common scene. In this research, the TREE-D-SEEK semantic annotation model and the CSDF ontology are discussed. The semantic annotation model is discussed further in the next chapter.

4.1.7 External tools

Certain external tools may be required in addition to the components outlined in the framework. These tools may be used for extracting feature from the content or to manually annotate 3D scenes. The TREE-D-SEEK framework has a wrapper class that can execute external programs synchronously and asynchronously. Some common classes that are used for both indexing and searching are described next.

4.1.8 Classes and interfaces common in both indexing and searching

Based on the process flow described in Chapter III, the searching process may also involve preprocessing, feature extraction and descriptor generation for a query before the query can be matched to the stored index. The Indexer classes in the TREE-D-SEEK framework encapsulate these processes and are shown in Figure 23.

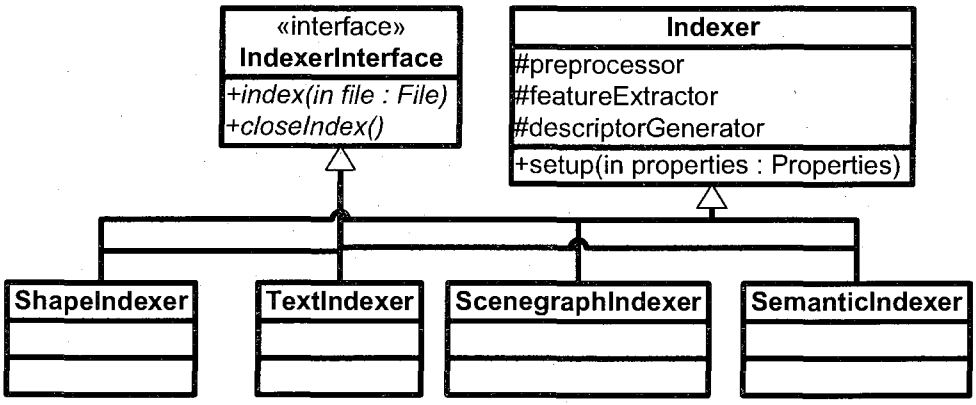


Figure 23: Indexers.

The preprocessor classes are responsible for primarily parsing the files which may be encoded in different formats. For shape based retrieval, preprocessing may involve model registration as discussed in Figure 24.

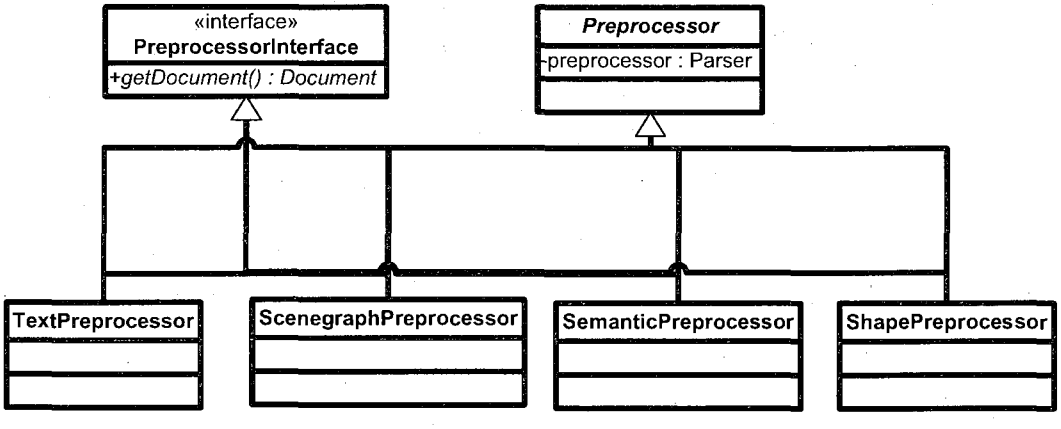


Figure 24: Preprocessors.

The FeatureExtractor classes are responsible for extracting features from the content. Each FeatureExtractor class contains an Analyzer class. The Analyzer class is responsible

for refining the features extracted from the content. For instance, for text based retrieval, the stemming, and keyword expansion is the responsibility of this class. The extractor classes are shown in Figure 25.

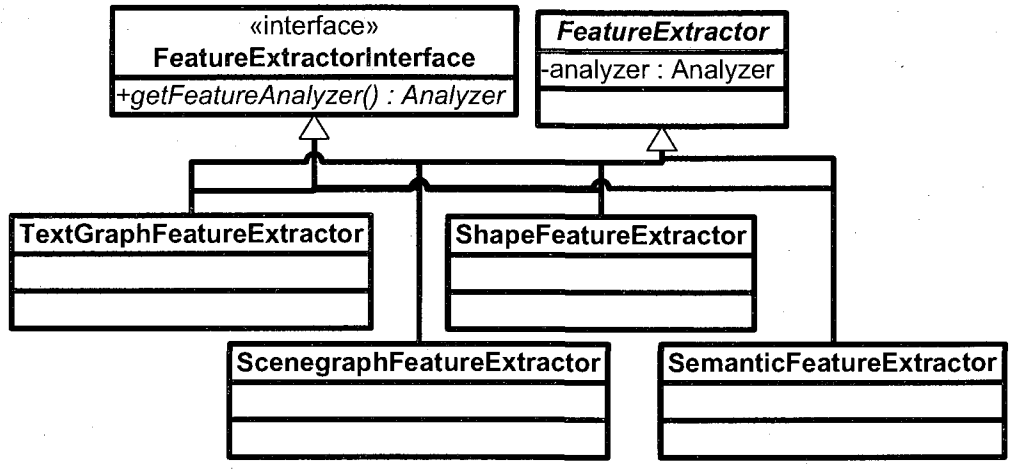


Figure 25: FeatureExtractors.

The DescriptorGenerator class is responsible for creating descriptors for extracted features. The DescriptorGenerator class can store the descriptors in the index if the process is invoked by the IndexerManager class. If the SearcherManager class invokes the process flow, then the descriptors are not stored but are returned from the DescriptorGenerator to the Searchers to be used further for querying the stored index.

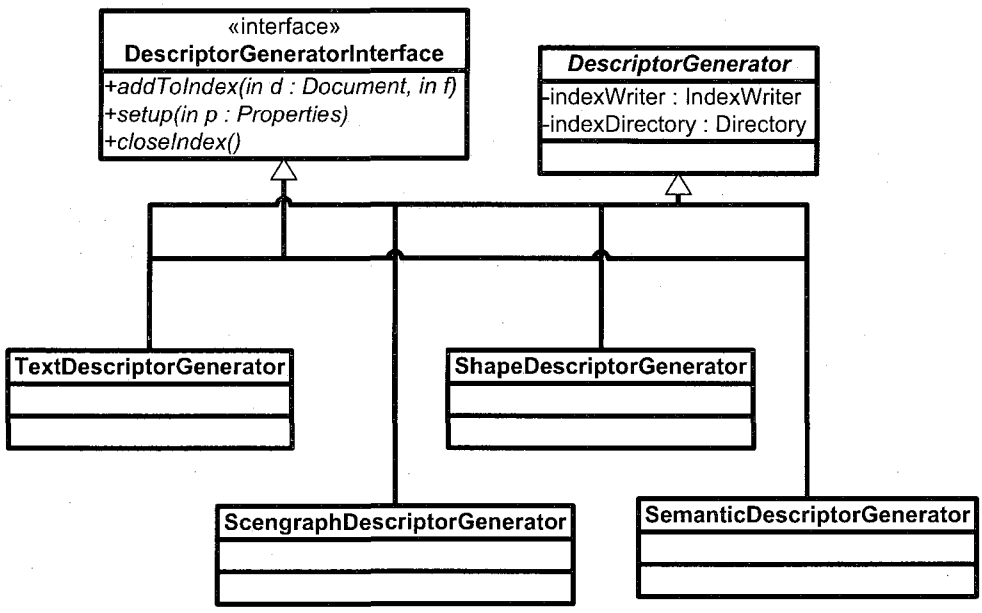


Figure 26: DescriptorGenerators.

For each type of document and query handled by the IndexerManager or the SearcherManager, a parser may be required for ingesting the content for feature extraction. The parsers implement the ParserInterface as shown in Figure 27.

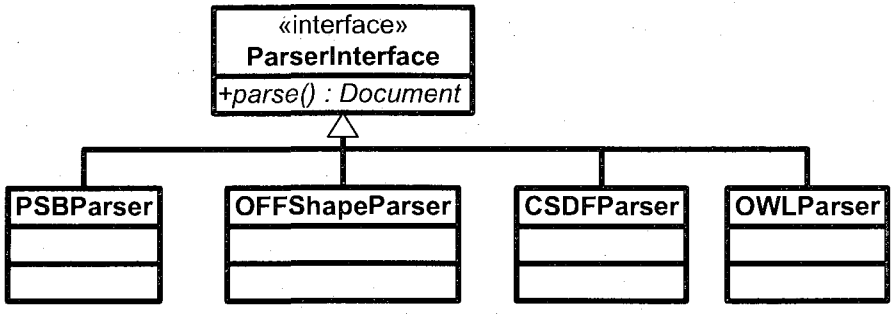


Figure 27: ParserInterface.

4.2 TREE-D-SEEK framework: A dataflow view

A dataflow view for the indexing process in the TREE-D-SEEK framework is shown in Figure 28. It is assumed that all content is already available in a file system accessible for indexing. The DirectoryTraverser is responsible for traversing recursively each file in the corpus and based on the file extension is sent for indexing to the IndexerManager. The IndexerManager is responsible for delegating the indexing of the content to its respective indexer. If shape based indexing is required then the file is sent to the ShapeIndexer.

Before the document is sent to either of the ScenegraphIndexer, ShapeIndexer or Semantic Indexer, the IndexerManager is responsible for sending the file to a CSDFFramework instance, wherein a CSDF scene definition is created. Next, the CSDF scene definition is sent to the CSAM component and to individual indexers. The CSAM component is responsible for also creating an OWL/RDF based annotations from the CSDF scene definition. The process of the annotation of the common scene definition may require human intervention. As mentioned previously, the CSAM module has not been fully implemented and at the time of writing this dissertation is not capable of fully formulating a formal specification for the common scene definition. In this research, a semantic annotation model for annotating 3D scenes for the CSDF framework and the TREE-D-SEEK framework is proposed.

Each indexer component has a preprocessor, feature extractor and descriptor generator component for creating feature descriptors for each file. Once the descriptors are created, they may be stored in the corresponding index.

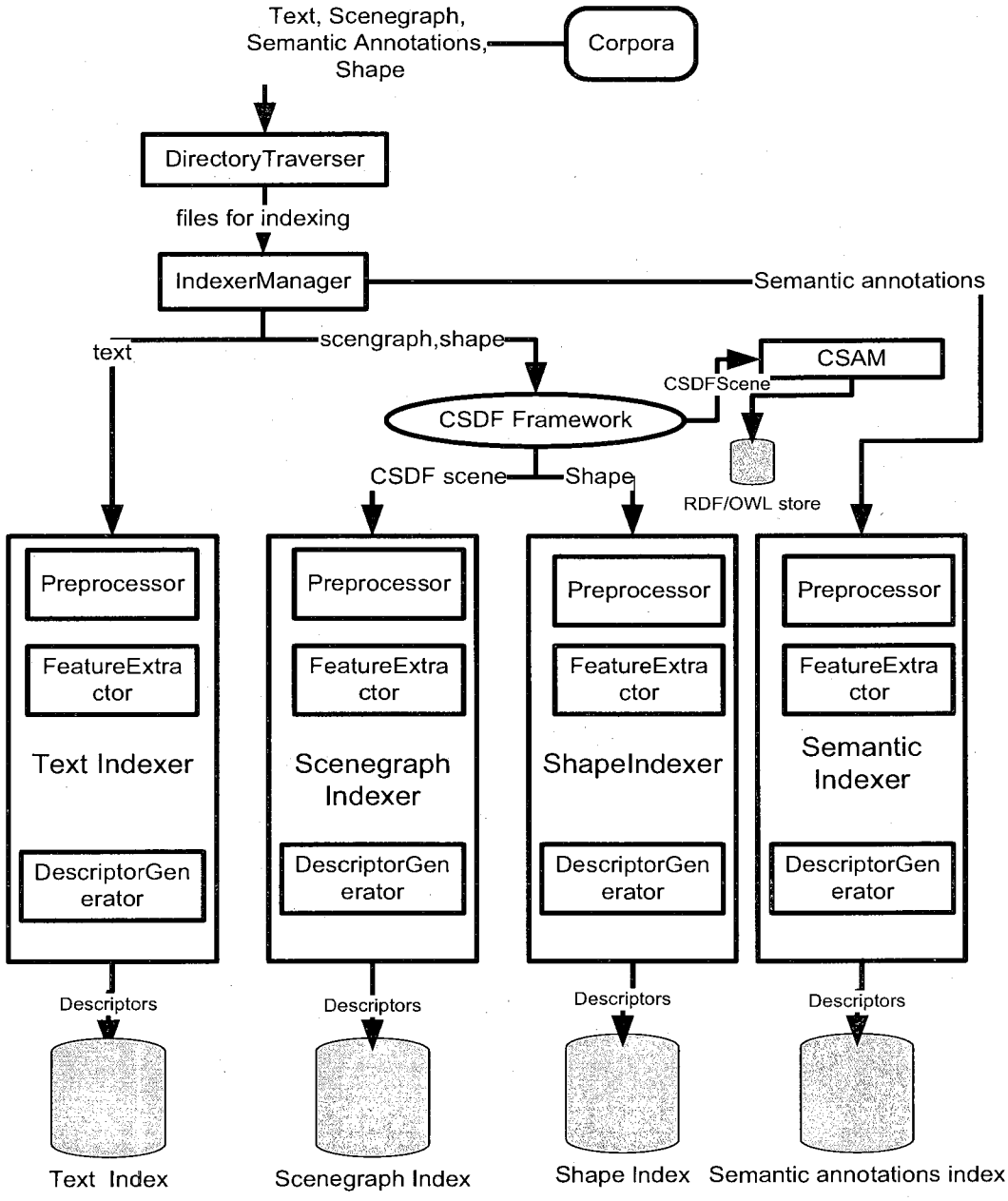


Figure 28: Indexing components and data flow in the TREE-D-SEEK framework.

The data flow in the searching phase of the TREE-D-SEEK framework is shown in Figure 29.

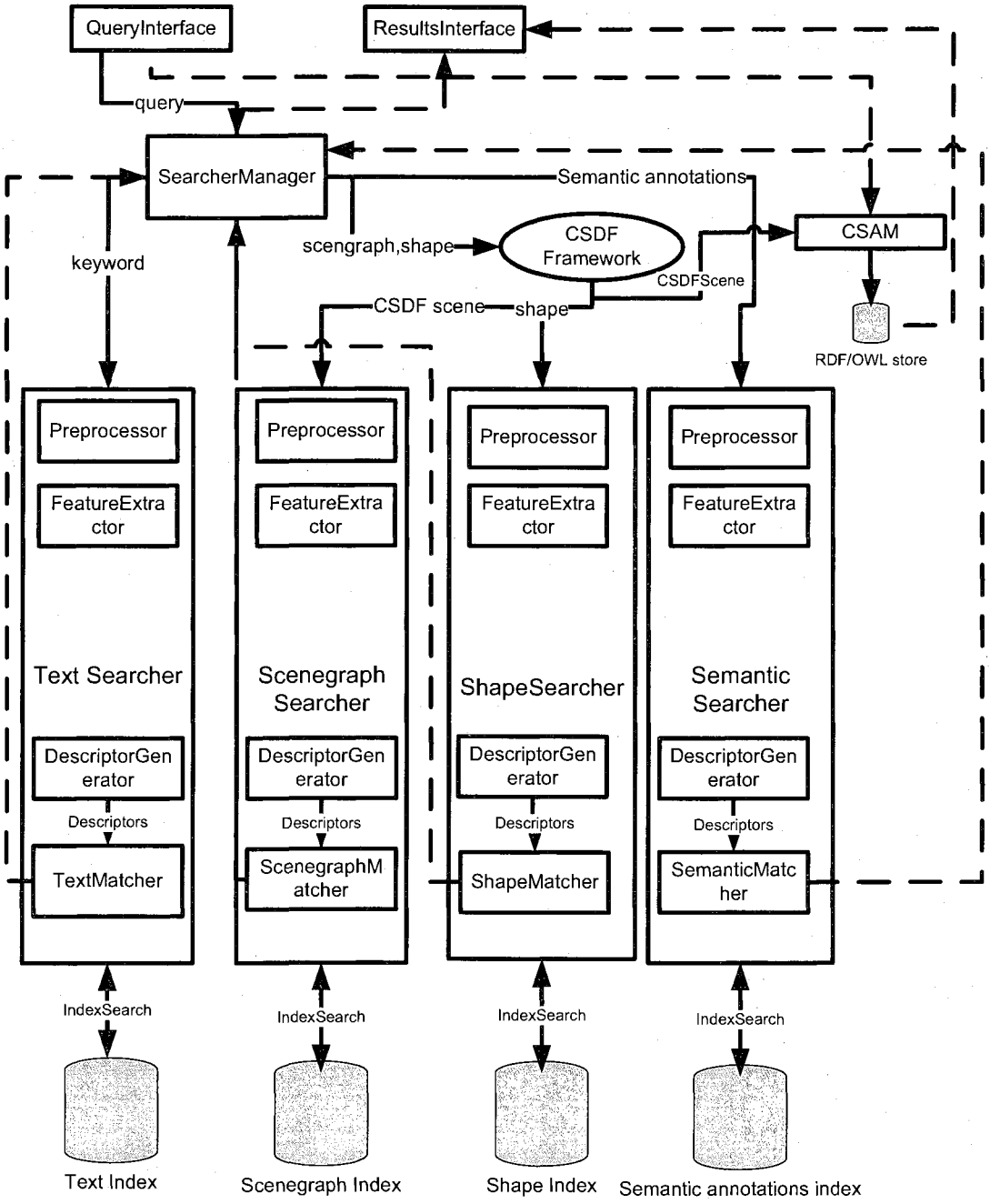


Figure 29: Dataflow in searching process.

4.3 TREE-D-SEEK framework implementation

As shown in Figure 30, the TREE-D seek framework is built on top of the Apache Lucene Information Retrieval Library [77]. From the developer's perspective, the framework uses the façade design pattern [78].

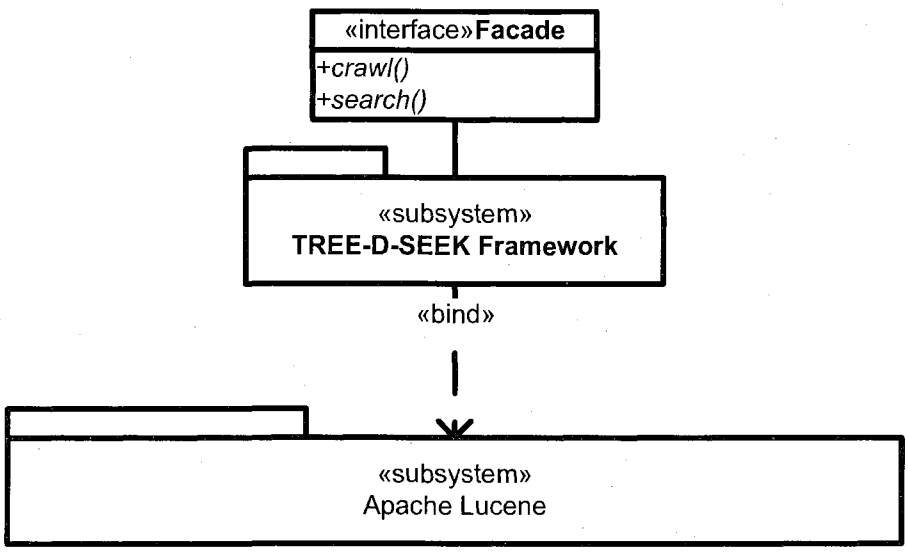


Figure 30: Software architecture: an implementation perspective.

Next, the implemented algorithms for indexing and matching text, scenegraph, shapes and semantic annotations are discussed.

4.4 Retrieval based on text

The implemented retrieval uses the Term Frequency Inverse Document Frequency (TFIDF) weighing strategy discussed in Chapter II. Documents are represented as vectors in a multidimensional Euclidean space where each distinct term is a dimension in this space. Each term is assigned a numeric weight to indicate the ability of the term to act as

a descriptor for the document and will typically have different weights in different documents. The weights assigned to terms in a document can represent the coordinates of the document vector in the Euclidean space. The corpus or collection of documents may be represented by using a document by term matrix where each row is a document and each column is a term and an entry at i^{th} row and j^{th} column indicates the weight of term j in document i. The most common weighing scheme used to assign weights to terms in documents is the Term Frequency Inverse Document Frequency (TFIDF) weighing scheme.

The Term frequency (t_f) is the frequency of occurrence of a given term in a document and is a local document specific statistic. Inverse Document Frequency (IDF) maps the frequency of the occurrence of a term over the entire corpus. IDF is defined as $\ln(\frac{N}{n})$ where N is the total number of documents in the collection and n is the number of documents containing the term. IDF is zero if a particular term appears in every document. This indicates that the term may not be a good descriptor for the document or any document in the collection.

In summary, in the TFIDF weighing scheme, heavier weights are assigned to terms that occur frequently within a document and do not occur too frequently in other documents or heavier weights are assigned to terms that occur moderately within a document and over documents in the collection. The similarity measure used to compare a query to each

document is the cosine angle between vectors, i.e. for two vectors $\vec{v1}$ and $\vec{v2}$, the cosine angle is shown below.

$$\cos\theta = \frac{v1.v2}{\|v1\|.\|v2\|} \tag{4}$$

There can be several potential text sources for retrieving 3D scenes. 3D scenes may have relevant filenames. Occasionally, these file names are alphanumeric indicating version number. Furthermore, some file names may have 3D as a suffix to them. As a consequence, the file names may need to be normalized. If 3D scenes are web based, then URL paths, hypertext, web page titles, and etc. may be potential text sources. The TREE-D-SEEK framework provides an HTML parser to parse relevant source and index the content. The TREE-D-SEEK framework provides an interface to the developer which may be extended to use any parser of the developer's choice. Stopping is done based on the SMART classification [79] of stop words. Stemming is done using the Porter Stemmer [80]. Keyword expansion is done using Wordnet [81].

4.5 Retrieval based on scenegraph matching

In the scenegraph retrieval strategy implemented in the TREE-D-SEEK framework, two descriptors are generated for every scenegraph. The first descriptor represents any metadata present in the scenegraph. For example, metadata nodes such as the *WorldInfo* node in VRML or the *Meta* node in X3D contain human readable metadata for the 3D scene. Also, technologies such as VRML and X3D allow the reuse of scenegraph nodes in a 3D scene by defining a node, giving it a name and reusing the node by reference to

the original node. This may be done in VRML using DEF and USE. Such nodes are also potential sources for descriptor generation. Words may be expanded using WordNet and then stored in the index. Scenegraphs are also indexed based on texture file names, diffusive color and other material properties. The second index created is solely for matching the structure of the scenegraphs. Two algorithms have been implemented for scenegraph structure indexing and matching.

4.5.1 Levenshtein distance

The *Levenshtein* distance edit distance algorithm [35] is implemented in the TREE-D-SEEK framework for indexing and matching scenegraphs based on structure. This is an algorithm that uses a dynamic programming technique to compare two labeled trees based on deleting, inserting and relabeling nodes. The scoring function is given by

$$1-\left(\frac{editDis\tan ce}{\min(queryLength, storedIndexLength}\right) \tag{5}$$

Where *edit Distance* is the reciprocal of the number of steps required by the Levenshtein Algorithm to match the two strings, *queryLength* is the length of the query string and the *storedIndexLength* is the length of the stored index.

As mentioned previously, the scenegraph of a 3D scene is a directed acyclic tree. The nodes in the acyclic tree represent 3D objects and associated behavior present in the 3D scene. The steps involved for creating a descriptor are shown in Figure 31.

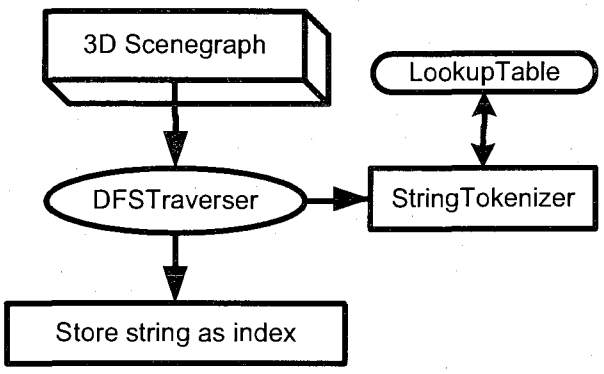


Figure 31: Scenegraph descriptor creation.

A lookup table is first created that contains each node type that may appear in the scenegraph along with a corresponding string identifier. The value of the identifier may be an integer based on automatically parsing the schema related to the scenegraph. A scenegraph tree is traversed using a depth-first algorithm. At the start of the algorithm, a string descriptor representing the structure of the scenegraph is initialized as empty. As each node is visited, a string identifier corresponding to the visited node is concatenated to the end of the string descriptor. If the node has a unique named attribute/label, it is also added to the text index. The string obtained after the traversal of each node in the scenegraph is the index/descriptor for the entire scene and is stored in the index.

For a query scenegraph, the above process is repeated. Once the string descriptor is created, the descriptor is matched with existing scenegraph descriptors using the Levenshtein distance algorithm. The other algorithm implemented is the tree isomorphism approach.

4.5.2 Tree isomorphism

TREE-D-SEEK also performs structure matching using combinatorial algorithms for tree matching. Each node is assigned a code based on a post order traversal. An example of this process is shown in Figure 32. The assignment of codes to each node is performed using a bottom up approach. The leaves of the tree are all given a code 1. Next, the parent of the labeled leaves is assigned a code that is a concatenation of the count of the total number of children+1 and the codes of the children sorted in ascending order of dimensions. Matching may be done by comparing the root nodes using the Levenshtein edit tree algorithm mentioned previously.

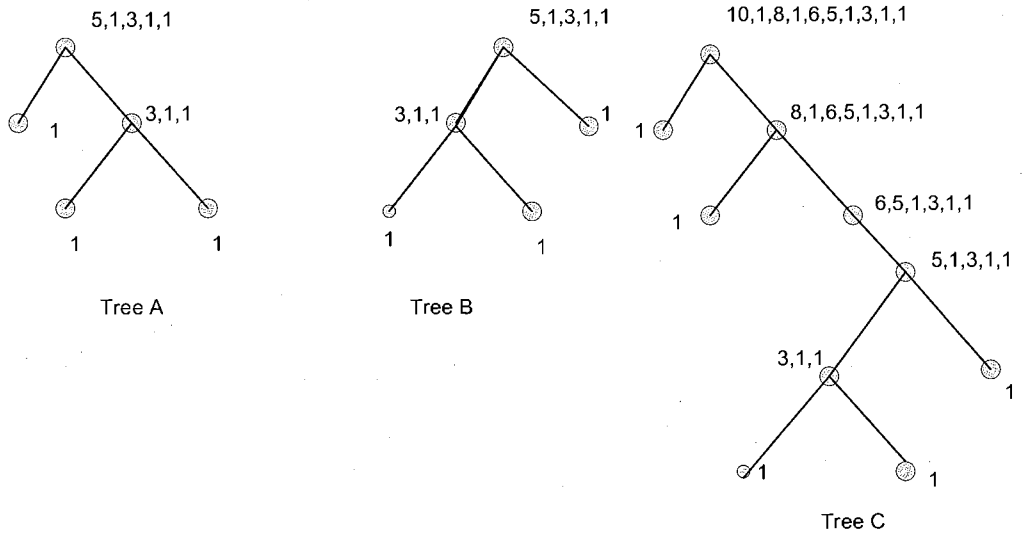


Figure 32: Isomorphism codes for trees.

4.6 Shape retrieval

The TREE-D-SEEK framework is capable of retrieving content based on matching the geometry of 3D objects. The proposed algorithm can only be used on 3D scenes that

contain isolated or single 3D models. The algorithm used is D2 on voxels [82]. The objective is to calculate the distribution of distances between voxels and comparing the histograms. The basic idea is obtained from the D2 algorithm for surface points given by Osada [43]. Since distance calculations are invariant to translation and rotations, these descriptors are easy to compute and matching is fast. The distance distribution function at a given distance d is given by

$$D2(d) = \frac{\left|\{\forall p, q \; \varepsilon \; P \text{ where } \|p-q\| = d\}\right|}{|P|^2} \tag{5}$$

where, P is the point set, ||p-q|| is the Euclidean distance between point's p and q and |P| is the number of points in P. The dissimilarity distance (for matching) is given by

$$\text{Diss\_D2}(\text{Histogram}_p - \text{Histogram}_q) = \sum_{i=1}^{n} \left| v_{pi} - v_{qi} \right|. \tag{6}$$

A pictorial representation of the data flow is shown in Figure 33.

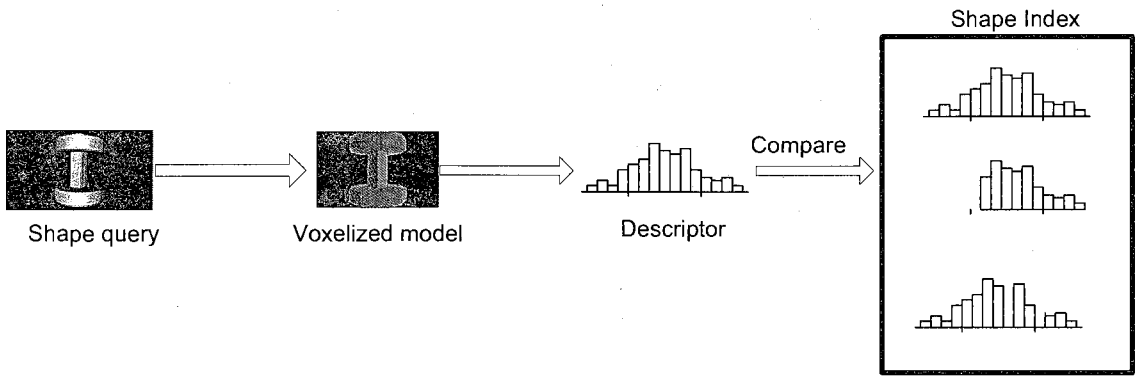


Figure 33: Shape matching.

The query model is first voxelized into a 256x256x256 voxel grid. Voxelization is performed using external tools [83]. Then the descriptor calculations are performed on

the model using (5). The query descriptors are then compared with the stored descriptors (6) in the shape index of the TREE-D-SEEK framework.

4.7 Semantic annotations retrieval

It is assumed that semantic annotations for 3D scenes are based on ontologies and are stored separate from the scene. Furthermore, for the purposes of this research, the ontologies and annotations must be based on the OWL formalism. A brief description of the OWL language is described in the Appendix. Again, for the purposes of this research, it is assumed that annotations are represented as individuals from the OWL formalism and both annotations and their corresponding ontologies are available for indexing.

As mentioned previously, current approaches to comparing ontologies is to compare the lexical and structural similarities of the underlying ontologies [76]. In lexical matching, ontologies are matched using any textual label, class name, property name, individual name, etc. In structural matching, the strategy is to represent ontologies using graph formalism and to perform graph matching for comparing ontologies.

In the TREE-D-SEEK framework, and as proof of concept, a lexical indexing and matching algorithm has been implemented. This involves using Wordnet based keyword expansion and TFIDF on natural language elements such as the comment nodes of the ontology. Classes and labels are first tokenized by removing any non alphabet symbols and then separated into individual words if camel case is detected. Then, the individual words are expanded using Wordnet. The obtained descriptors are then stored into the

index using the TFIDF weighing approach. In addition, the class names and labels are also treated as distinct strings that may be matched based on the Levenshtein distance algorithm mentioned previously. A weighted matching scheme based on the TFIDF and the Levenshtein distance algorithm may be used for the overall scoring of the annotation file.

Chapter V

TREE-D-SEEK SEMANTIC ANNOTATION MODEL AND CSDF ONTOLOGY

As mentioned in Chapter IV, the rationale and objective for providing a formal specification for a 3D scene indexed using the TREE-D-SEEK framework is to support richer queries, improved retrieval and navigation within the 3D scene. From a framework and 3D retrieval perspective, this strategy is not derived based on any commonality in process flows of existing 3D retrieval solutions.

The Common Scene Annotation Modeler (CSAM) has been mentioned in the previous chapter. The CSAM module accepts the common scene definition from the CSDF framework and is responsible for semi-automating the generation of a formal specification of the 3D scene based on the TREE-D-SEEK semantic annotation model.

At the time of writing this dissertation, the implementation of the CSAM module is not complete. The CSAM module is not capable of fully specifying a common scene definition obtained from the CSDF framework. The semantic annotation model based on the CSDF framework is presented.

5.1 CSDF ontology

In general, the focus of 3D modelers is primarily on the geometric modeling and rendering aspect of 3D scenes. The semantic annotation of 3D scenes is usually not part

of a formal design process in 3D scene development. Indeed, most 3D authoring tools provide limited capabilities for specifying semantic annotations for 3D content. In this chapter, a semantic annotation model for 3D scenes is proposed. The TREE-D-SEEK annotation model is based on an ontology derived from the CSDF framework. The CSDF framework is discussed in [72]. It consists of a superset of existing 3D technology along with provisions for extensibility to include new 3D technologies. The top level view of a section of the CSDF ontology is shown in Figure 34. Many of the classes present in the CSDF ontology is based on the software classes in the CSDF framework. For instance, the CSDFGeometry class exists both as a Java class in the CSDF class and an OWL class in the CSDF ontology.

The ontology also provides concepts that can be used to store descriptors obtained from the indexing strategy described in Chapter III. A 3D scene consists of one or more scenegraphs. Each scenegraph has a collection of one or more *GroupHolder*. A GroupHolder may contain a logical grouping of 3D objects. A GroupHolder may have a mapping to a real world entity. For instance four walls can be grouped to a room. The *hasRealWorldMapping* property maps a 3D object to its equivalent real world object name.

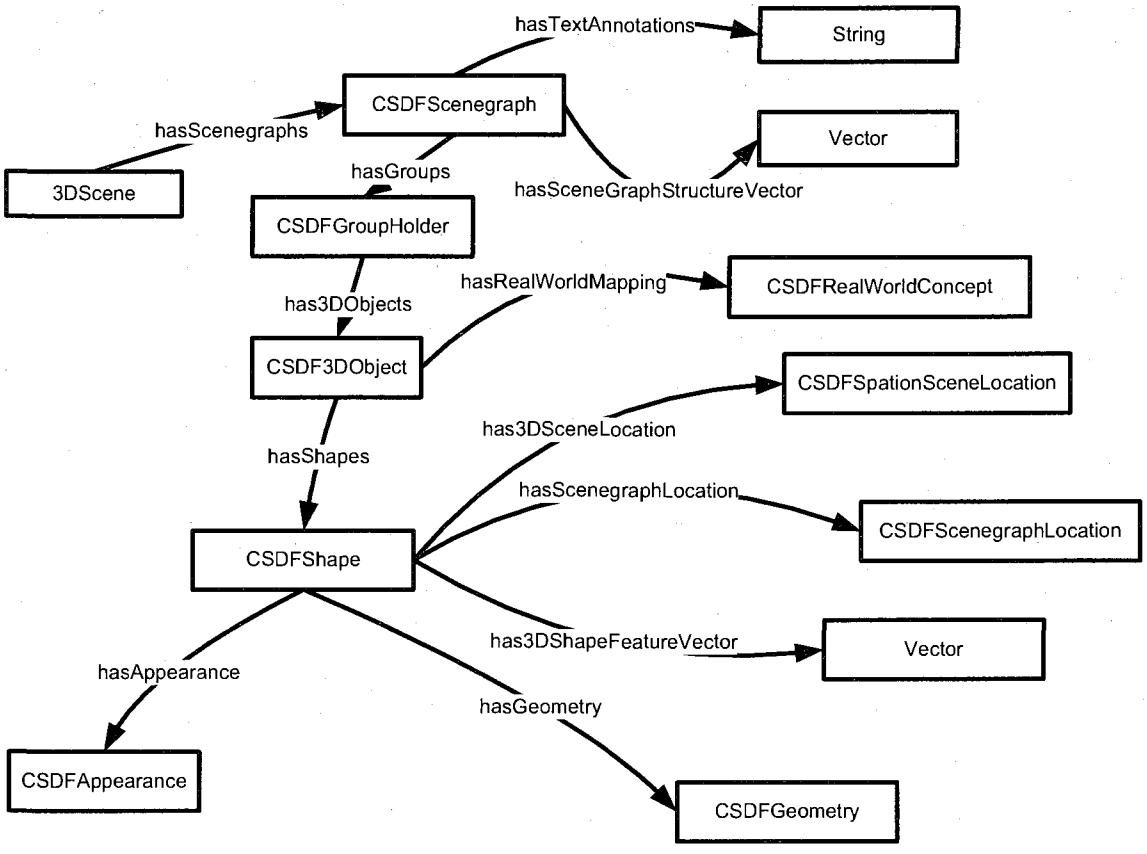


Figure 34: An incomplete representation of the CSDF ontology.

The *hasPart* property allows 3D objects to be segmented into subparts which may or may not have a real world mapping. A GroupHolder may contain other GroupHolders by using a transitive property called *controls.* A GroupHolder may consist of one or more 3D objects. The CSDFGroupholder concept taxonomy is shown in Figure 35.

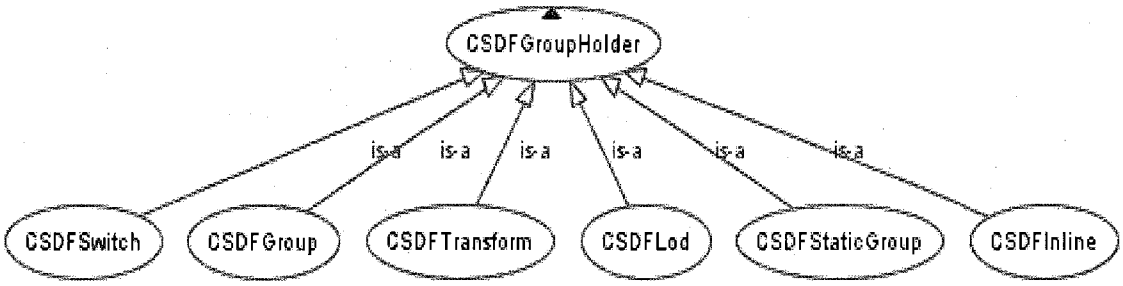


Figure 35: GroupHolder taxonomy.

The properties associated with the CSDFTransform concept is shown in Figure 36.

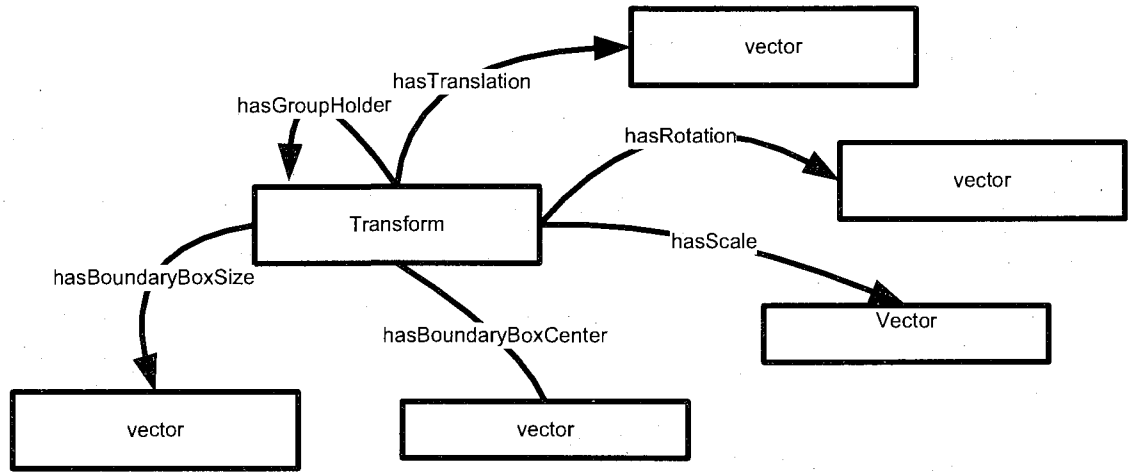


Figure 36: CSDFTransform properties.

The 3D object may be an aggregation of 3D shapes. Each 3D shape has geometry and an appearance associated with it. The Appearance concept specifies a material, color or texture associated with the shape. An incomplete representation of the Appearance concept is shown in Figure 37.

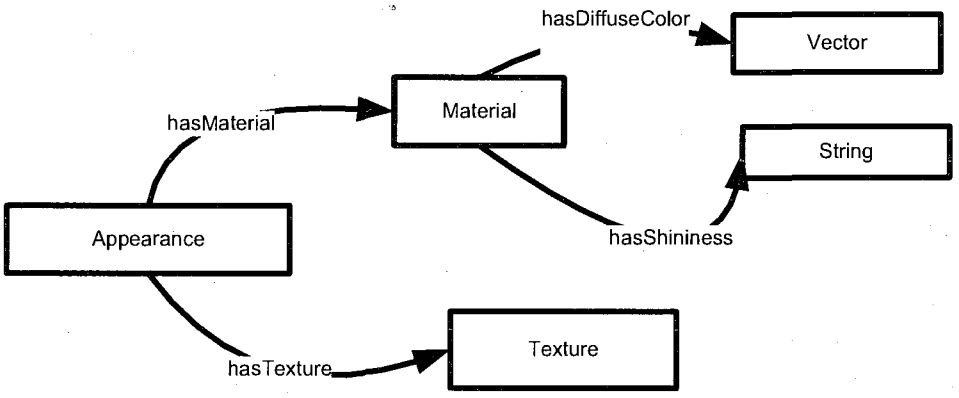


Figure 37: Appearance concept.

The CSDFGeometry concept taxonomy is shown in Figure 38.

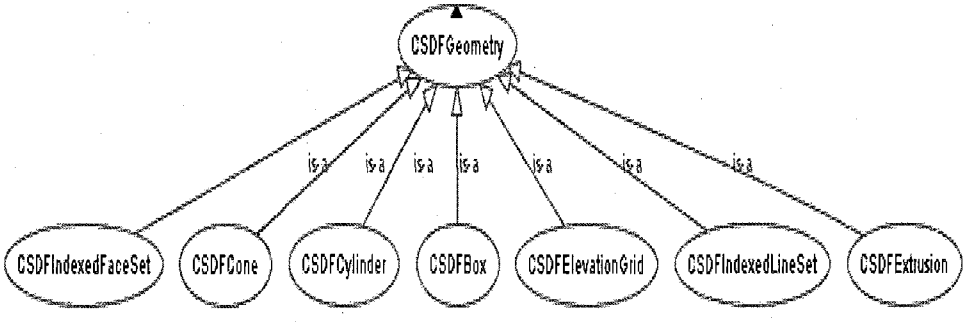


Figure 38: CSDFGeometry taxonomy.

Some of the properties associated with a CSDFCylinder are shown in Figure 39.

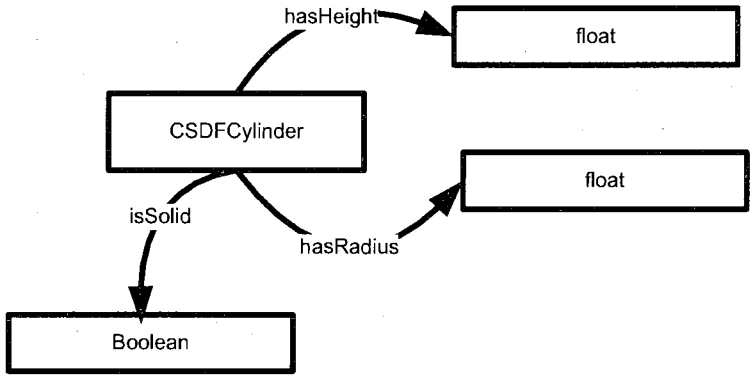
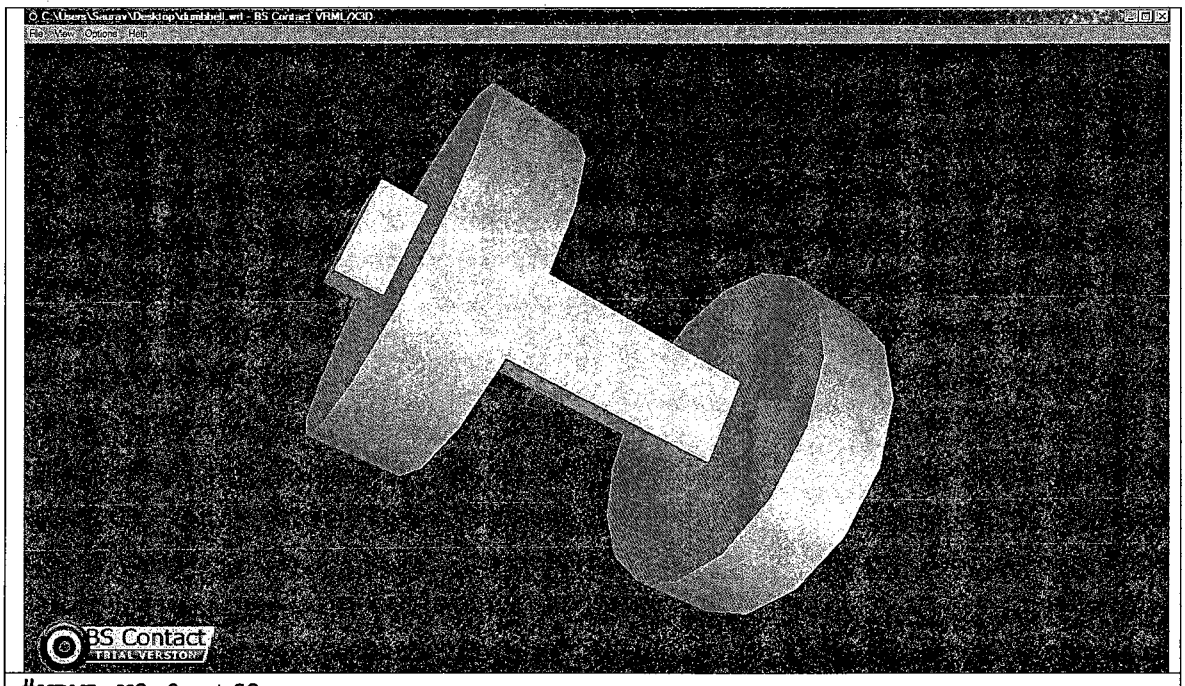


Figure 39: CSDFCylinder properties.

An example of a 3D scene in VRML 2.0 is shown in Figure 40.




```
#VRML V2.0 utf8

Group {
  children [
#-------------------cylinder---------------------------------
      Shape {
            appearance DEF MY Appearance {
                  material Material {
                              diffuseColor 1.0 1.0 1.0
                  }
```

```
        }
            geometry Cylinder {
                radius 1
                height 0.5
                side TRUE
                top TRUE
                bottom TRUE
            }
    }
#-------------------------box-------------------------------
    Transform {
          Translation 0 1 0
          children [
                Shape {
                      appearance USE MY
                       geometry Box {
                       size 0.5 3 0.5
                }
          }
          ]
    }
#----------------------cylinder-----
    Transform {
      translation 0 2 0
        children [
                Shape {
                      appearance USE MY
                      geometry Cylinder{
                      radius 1
                      height 0.5
                      side TRUE
                      top TRUE
                      bottom TRUE
                       }
                }
        ]
      }
    ]
   }
```

Figure 40: Dumbbell in VRML 2.0.

The corresponding annotation of the above 3D scene based on the CSDF ontology is shown in Figure 41.

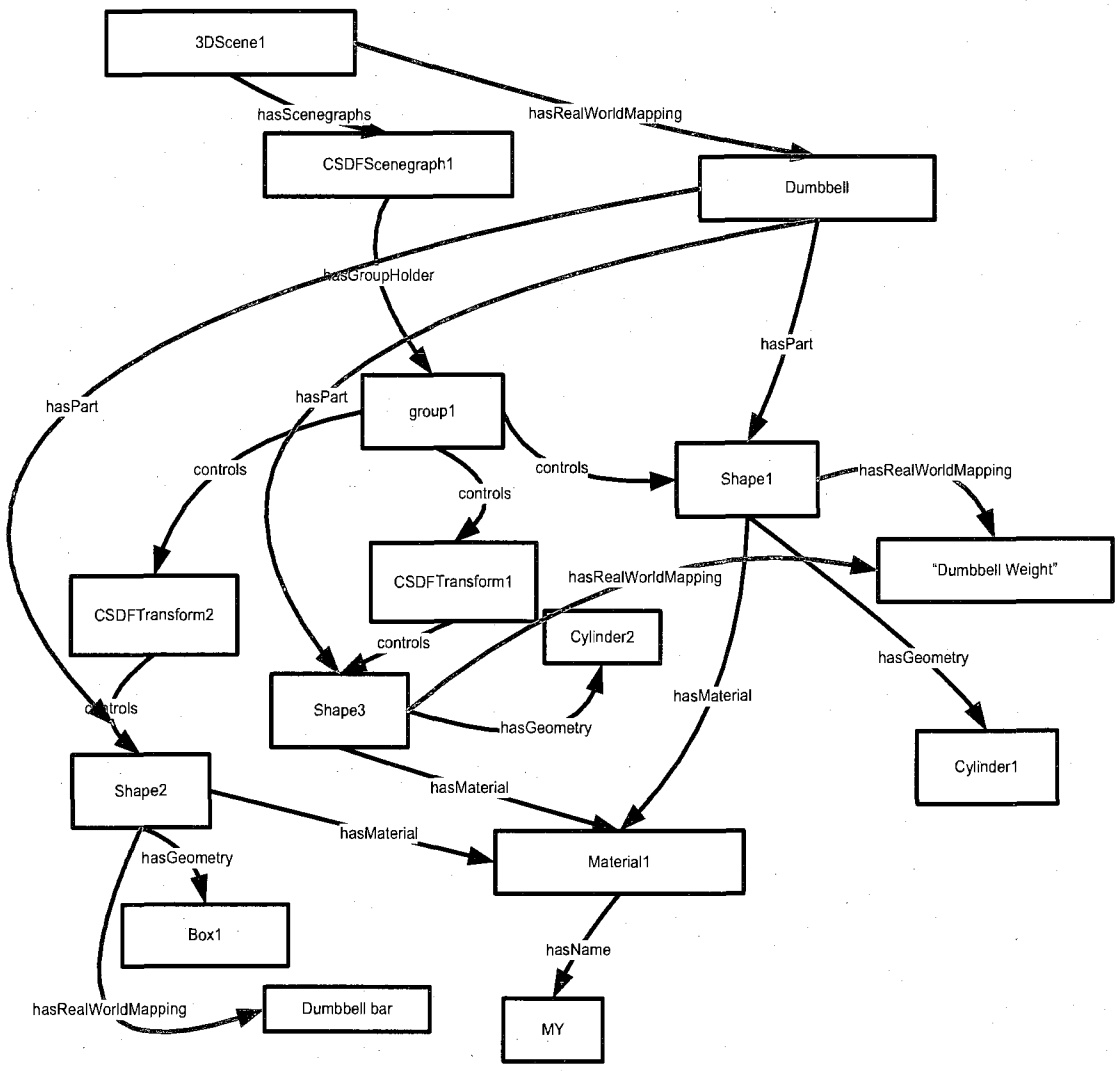


Figure 41: 3D scene annotation using the CSDF ontology.

The CSDF ontology supports the indexing mechanism described in this research. The *has3DShapeFeatureVector* and *hasMatching* properties may be used to specify shape descriptors and matching functions used for indexing and retrieving 3D shape. The *hasScenegraphFeatureVector* may be used to store the scenegraph feature vector. Some of the potential benefits of using the annotation scheme in the TREE-D-SEEK framework are as follows:

1. Richer queries are possible such as "retrieve all 3D objects present in the scene whose color is blue", "retrieve all 3D object that have a box as a part", etc.
2. A directed, rooted, connected, acyclic scene-graph such as the VRML/X3D scenegraphs is limited in its ability to express semantic relationships. Two limitations mentioned in [64] are the difficulty of expressing a shared object from a logical level and the difficulty in expressing the semantics associated with the functionality or property of an object in the scenegraph. The CSDF ontology is not acyclic and can represent shared objects. It also provides the *hasFunction* property that may be used to indicate the real world functionality of an object.
3. Many of the scenegraph objects, properties and relationships can be automatically obtained by using the TREE-D-SEEK framework. Relationships that contain child nodes with two or more parent nodes will need to be specified manually.
4. The CSDF ontology may be used to specify a language independent high level specification of a 3D virtual scene or world. This specification may be used to populate the CSDF framework, thereby allowing rapid prototyping of 3D virtual worlds in any 3D format and technology supported in the CSDF. This capability is not the focus of this work and has not been addressed.
5. This strategy of coupling a search engine framework with a rapid prototyping framework supports the notion of 'search' as being part of a

formal process in 3D scene authoring and in the rapid prototyping of 3D virtual worlds.

Chapter VI

IMPLEMENTATION, EXPERIMENTS AND DISCUSSIONS

The primary objective of this research is to design and implement a framework for retrieving 3D content. Search engine developers may use, modify or extend the framework to provide desired solutions for retrieving 3D content. The framework supports 3D retrieval based on indexing and matching syntactic metadata, scenegraph structure, shape and semantic annotations. Several indexing and matching algorithms have been implemented as proof of concept that the software framework is indeed capable of encapsulating descriptor based 3D indexing and matching strategy for disparate information sources. A semantic annotation model is proposed in the TREE-D-SEEK framework to support richer queries and improved search within 3D scenes. At the time of writing this dissertation, the framework is not fully capable of indexing 3D scenes based on the proposed semantic annotation model. This is because the CSDF ontology does not contain all classes corresponding to the CSDF software classes in java.

In this chapter, several experiments are conducted to test the retrieval effectiveness of the indexing and matching algorithms implemented as part of the framework. The goal is to show that such experiments are possible based on the use of this framework and not necessarily to verify the performance of individual implementations of algorithms. The rest of the chapter is organized as follows. First, details of the implementation of the TREE-D-SEEK framework are discussed. Next, supported query expressions for each

type of content are presented. This is followed by the result of some experiments conducted to test the retrieval effectiveness of the implemented indexing and matching algorithms. Finally, a discussion of the potential use of a search system that uses the framework from an end user's perspective is provided.

6.1 Implementation details

The TREE-D-SEEK framework was written in Java 6 and is built on top of the Apache Lucene [77] framework. The framework provides capabilities to store descriptors as part of the Lucene index or as entries in a MySQL database. The framework provides the capability to make system calls to execute external programs. Text, scenegraph, shape and semantic annotations based indexing and matching algorithms are implemented as part of the framework. For shape based retrieval, 3D models were voxelized using [84]. For creating shape-based descriptors, an external descriptor generator program using C++ was created. The framework calls this descriptor generator program during shape indexing and searching to perform shape based retrieval.

6.1.1 Administration

For the purposes of this research, information sources used for testing the framework contained plain text, scenegraph, shape and semantic annotation in OWL. The framework also has built in parsers for HTML and XML. To administer the indexing and searching processes in TREE-D-SEEK, the framework provides control files for indexing and searching. An example of an indexing administration file and an example of a searching administration file are shown in Figure 42 and Figure 43 respectively.

```
//TREE-D-SEEK framework indexing file
TextIndexer = false
TextExtension=txt
TextParser= parsers.TextFileParser
TextCommandLineFeatureAnalyzer=Analyzers.SynonymAnalyzer
TextIndexDirectory= c:\\phd\\demo\\textretrieval\\textindex


ScenegraphIndexer=true
ScenegraphExtension=x3d
ScenegraphFeatureAnalyzer=Analyzers.SynonymAnalyzer
ScenegraphIndexDirectory=c:\\phd\\demo\\scenegraphretrieval\\
scenegraphindex



ShapeIndexer=false
ShapeExtension=xml
ShapeFeatureExtractor=d2



SemanticIndexer=false
SemanticExtension=owl
SemanticFeatureAnalyzer=org.apache.lucene.analysis.standard.StandardAnaly
zer
SemanticIndexDirectory=c:\\phd\\demo\\SemanticRetrieval\\semanticindex
```

Figure 42: Indexer-administration file.

The *indexer-administration file* provides the capability to select the particular directory that contains the target corpora and also to select a directory to store the index if required. The other options are that descriptors generated may be stored in the Random Access Memory (RAM) or a MySQL database. The file also provides the capability to control the indexing process wherein a particular type of indexing may be switched on or off. For example, in Figure 42, shape, semantic and text indexing is switched off. If more than one type of indexing is turned on, the document in the corpora is indexed sequentially based on the order in which the type of indexing is specified in this file.

```
# setup file for search component of TREE-D-SEEK

TextCommandLineSearcher = false
TextCommandLineFeatureAnalyzer=Analyzers.SynonymAnalyzer
indexDirectory=C:\\phd\\demo\\textretrieval\\textindex



TextSearcher = false
TextExtension=txt
TextParser= none
TextFeatureAnalyzer=org.apache.lucene.analysis.standard.StandardAnalyzer
TextIndexDirectory= c:\\phd\\phdData\\index\\textIndex
Boost=

ScenegraphSearcher=true
SearchMode=true
ScenegraphExtension=x3d
ScenegraphFeatureAnalyzer=Analyzers.SynonymAnalyzer
ScenegraphIndexDirectory=c:\\phd\\demo\\scenegraphretrieval\\scenegraphindex
StructuralPrecision=1.0f
StructuralBoost=0.5
TextBoost=0.5

ShapeSearcher=false
ShapeExtension=xml
ShapeFeatureAnalyzer=d2
ShapeIndexDirectory=c:\\phd\\demo\\scenegraphretrieval\\shapeindex
Boost=0


SemanticSearcher=false
SemanticExtension=owl
SemanticFeatureAnalyzer=org.apache.lucene.analysis.standard.StandardAnalyzer
SemanticIndexDirectory=c:\\phd\\demo\\SemanticRetrieval\\semanticindex
Boost=1.00f
StructuralBoost=1.00f
TextBoost=1.00f
```

Figure 43: Searcher-administration file.

The *searcher-administration file* provides capabilities similar to its counterpart- the indexer administration-file such as the capability of turning off a particular type of search for an information source or level. The file also provides the capability of weighing a

indexing or matching scheme. For example, in the scenegraph based retrieval method weights may be assigned individually to scenegraph content and structure so that the user can "boost" the significance of either scenegraph content or scenegraph structure to improve relevance during retrieval.

6.1.2 CSAM Implementation details

The TREE-D-SEEK semantic annotation model and the CSDF ontology were created using Protégé 4.0 Beta [85]. The ontology had been written using the OWL-DL formalism. The CSAM module uses the OWL API [86] to parse the target file to be indexed and to create a new OWL based TREE-D-SEEK semantic annotation file. The semantic annotation file contains only a mapping between the classes generated using the CSDF framework and the OWL classes of the CSDF ontology. To provide higher level semantics, manual annotation or a manual mapping to a higher level class/concept of domain ontology is needed.

6.1.3 CSAM current implementation status

The CSDF ontology has not been completed at the time of writing this dissertation. The current implementation status of the CSDF ontology is shown as Table 1. A checkmark indicates that the corresponding class in the CSDF ontology has capability to express all its associated object and data properties and axioms. At the time of writing this dissertation, the CSDF ontology consists of 58 classes.

Table 1: Available classes in CSDF ontology.

CSDF Framework Class	CSDF OWL Class
CSDFAppearance	✓
CSDFBox	✓
CSDFColor	✓
CSDFCone	✓
CSDFCoordinate	
CSDFCylinder	✓
CSDFDirectionalLight	
CSDFExternProtoDeclare	
CSDFField	
CSDFFieldObjRef	
CSDFFontStyle	
CSDFGroup	✓
CSDFImageTexture	
CSDFIndexedFaceSet	✓
CSDFIndexedLineSet	✓
CSDFMaterial	✓
CSDFNavigationInfo	✓
CSDFNode	✓
CSDFNormal	
CSDFOrientationInterpolator	
CSDFPixelTexture	
CSDFPlaneSensor	

CSDFPointLight	
CSDFPointSet	
CSDFPositionInterpolator	
CSDFProtoInstance	
CSDFProximitySensor	
CSDFRoute	
CSDFScene	✓
CSDFScript	
CSDFShape	✓
CSDFSphere	✓
CSDFSwitch	✓
CSDFText	
CSDFTextureCoordinate	
CSDFTextureTransform	
CSDFTimeSensor	
CSDFTouchSensor	
CSDFTransform	✓
CSDFUse	
CSDFViewpoint	

6.1.4 Supported query expressions

The TREE-D-SEEK framework provides direct support for the following types of query expressions for information retrieval. For scenegraph and shape-based retrieval, a user submits an entire 3D model/scene file as a query. For retrieval based on semantic

annotations, a user is also required to submit an entire OWL annotation file. TREE-D-SEEK supports several types of query expressions for text based retrieval. For text based retrieval, a user may provide a document that can be parsed into text using one of the parsers available in the TREE-D-SEEK framework. At the time of writing this dissertation, the framework supports parsing of Hyper Text Markup Language (HTML) documents and Extensible Markup Language (XML) documents. The developer can also create their own parser implementations for parsing the desired type of content.

In addition, the TREE-D-SEEK framework supports automatic handling of several textual query expressions. A user may enter a query expression containing keywords for searching. A query expression may contain Boolean operators such as AND, OR and NOT. A query entered may be a phrase. A query may have wild cards in it. A query may have a range within which a result should be obtained. The user may also extract text from other media and sources provided that a parser is available and useable by the TREE-D-SEEK framework. For text, the supported query type, an example of the supported query syntax, and the contents of matched document are shown as Table 2.

Table 2: Supported text query expressions.

Text query type	**Example of supported query expression**	**Documents retrieved will contain**
Boolean based queries	Lets tree AND seek	text containing both TREE and SEEK
Phrase queries	" TREE-D-SEEK rocks"	text contain the words TREE and SEEK

		juxtaposed with each other
Wild card queries	TREE*	Text that contains the word TREE followed by a string of characters
Range Queries	Date[031609 TO 031709]	a date between 031609 and 031709

6.2 Experiments

The indexing and matching algorithms implemented as part of the TREE-D-SEEK framework are tested. A test-bed consists of two corpora namely the Princeton Shape Benchmark and a TREE-D-SEEK corpus is used and created respectively. The TREE-D-SEEK corpus is a combination of manually collected set of documents pertaining to text, scenegraph, shape and semantic retrieval. As mentioned previously, the primary objective of this work is to create an open, extensible framework for retrieving 3D scenes. The experiments discussed in this chapter prove that the framework can indeed retrieve 3D scenes based on text, shape, scenegraph and semantic annotations.

The rest of this section is organized as follows. First, the test bed is discussed. Next, some timing results are presented for indexing content based on text, scenegraph, shape and semantic annotations. Next, retrieval effectiveness is calculated using average-precision/recall curves using a technique described in the appendices.

6.2.1 Test bed

6.2.1.1 Princeton Shape Benchmark (PSB)

The PSB consists of freely available 1814 3D models from the Web. The models in the PSB have been manually classified. The available categories are shown in Figure 44. The 3D models are stored using a polygonal surface format. The PSB is purely a shape benchmark. It does not support the evaluation of text or semantic annotation retrieval algorithms.

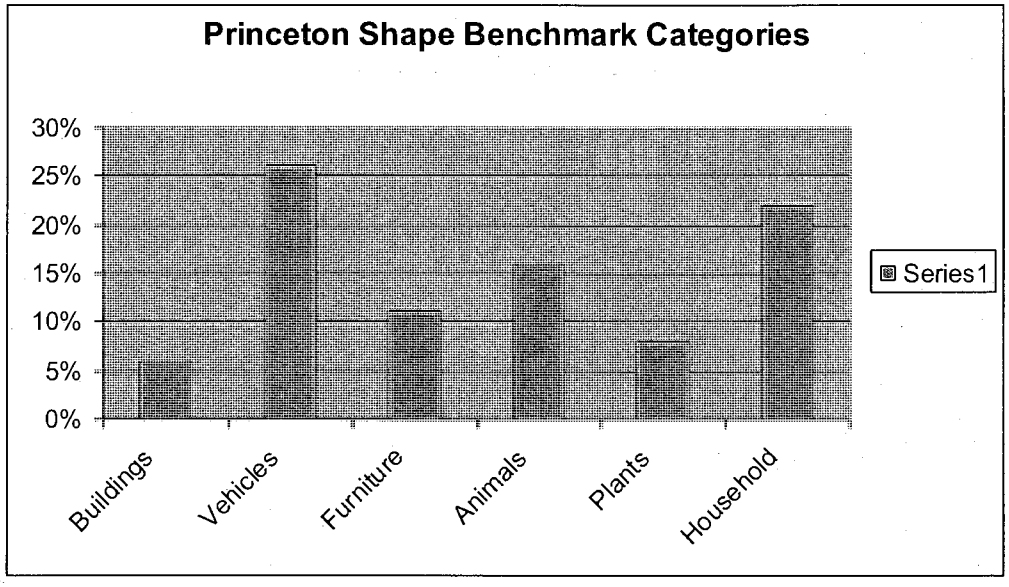


Figure 44: PSB categories.

6.2.1.2 TREE-D-SEEK corpus

Twenty 3D scenes were collected and manually classified. They were obtained from [87] and [88]. The 3D scenes were either VRML 2.0 or X3D based. The scenegraphs had structure and text that were used for retrieval. The corpus also contains twenty OWL

ontologies collected from the WWW. Each OWL file was populated with random individuals/instances. Each 3D scene in the corpus consists of basic shapes. The 3D scenes were manually classified in two groups. In one group, each 3D scene contained only 3D primitive shapes. Scenes in the other group contained the same primitive shapes with an axis rendered through though each shape.

6.3 Indexing times

The framework was used to evaluate the time taken by the framework for performing text, scenegraph, shape and semantic indexing. The test was conducted on a non dedicated machine for indexing with the specifications shown in Figure 45.

Hardware environment

CPU: Intel Core Duo at 1.73 Ghz

RAM: 2038 MB

Drive configuration: SCSI

Software environment

Lucene Version: 3.1

Java Version: 1.6

OS Version: Windows Vista Home

Location of index: file

Figure 45: Hardware and software configuration.

The TREE-D-SEEK corpus was used to obtain the graph shown in Figure 46. As expected, scenegraph retrieval with lookup took longer than the scenegraph indexing with no lookup, semantic indexing and text indexing. The scenegraph indexing with no lookup

uses the bottom up isomorphism strategy for indexing structure. The text corpus used in this experiment is from [89]. For shape indexing, the PSB was used. For each model, the average indexing time was 45.1 seconds.

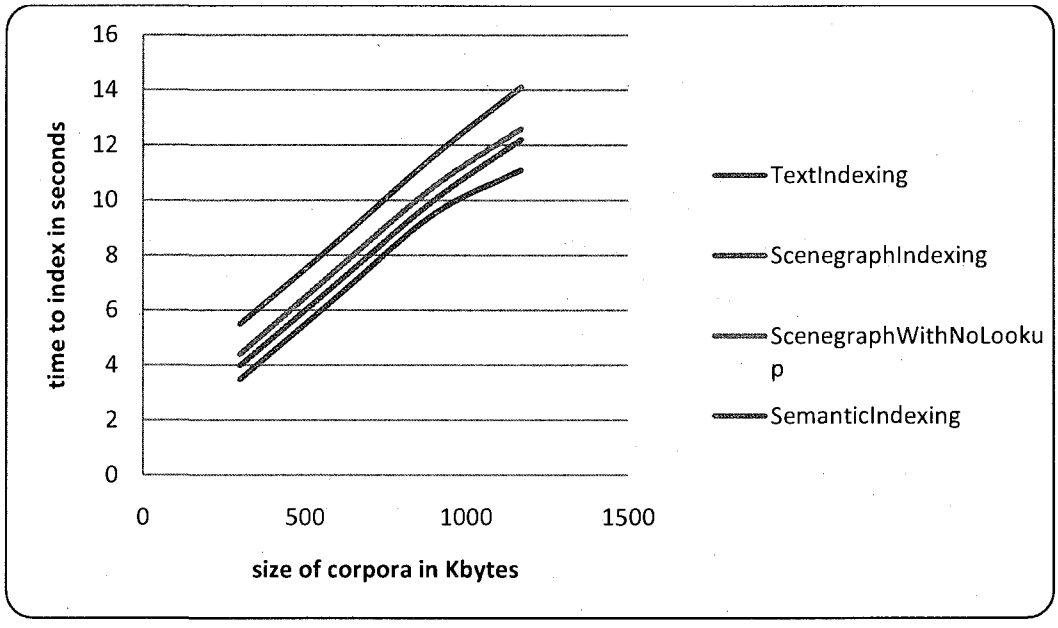


Figure 46: Indexing times.

6.4 Retrieval performance

Retrieval performance of the implemented indexing and matching strategies are discussed using precision and recall. A discussion on Information Retrieval (IR) metrics, recall and precision metrics and how the recall-precision graphs are obtained is discussed using an example in Appendix A1.

6.4.1 Text based retrieval

The TREE-D-SEEK corpus was used to test the TFIDF weighting scheme. Text was collected from [89]. The average precision/recall curve is shown in Figure 47.

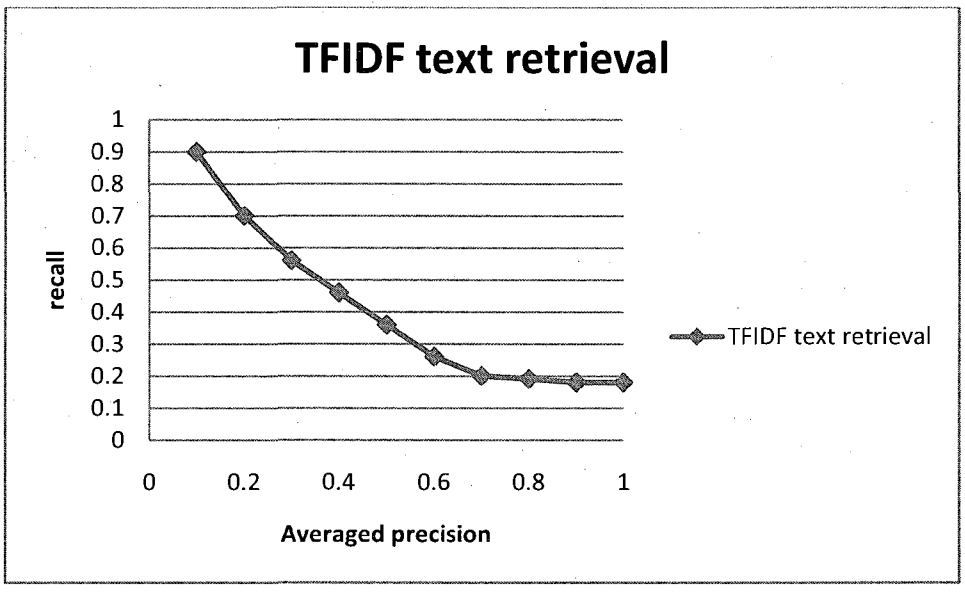


Figure 47: Average precision/recall for text retrieval.

6.4.2 Shape based retrieval

Since the PSB is a shape benchmark, D2 on voxels was used for shape matching. In, Figure 48, the average precision over recall curve for retrieval based on shape matching and scenegraph is presented. The curve presents precision recall values averaged over all models in the PSB. The PSB is purely a shape benchmark. Each model in the PSB consists of only polygon soups.

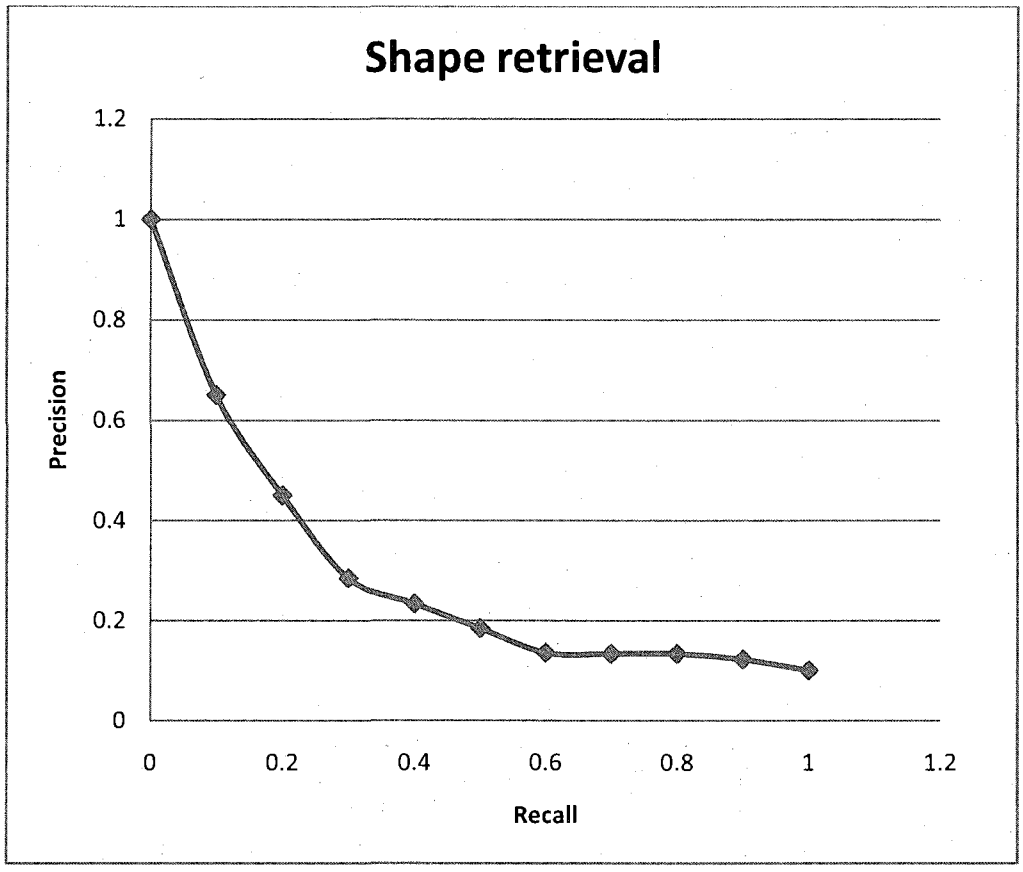


Figure 48: Averaged precision versus recall plot.

6.4.3 3D scenegraph retrieval

To test scenegraph retrieval, the TREE-D-SEEK corpus was used. The 3D scenes were manually classified into two groups-3D scenes that contain an axis rendered through the objects in the scene and 3D scenes that consist of basic geometric shapes only and no axis. The scenegraphs were indexed both for structure and content using the strategies discussed in chapter IV. Scenegraph retrieval was compared with 3D shape retrieval for this corpus, and the results are shown in Figure 49. For the given corpus, retrieval using scenegraph content and structure yielded better recall-precision than the shape matching

using D2 on voxels. This was because the distinction between the shapes with axis and shapes without axis was not as significant for the D2 on voxels shape indexing method as it was in scenegraph structure and content indexing.

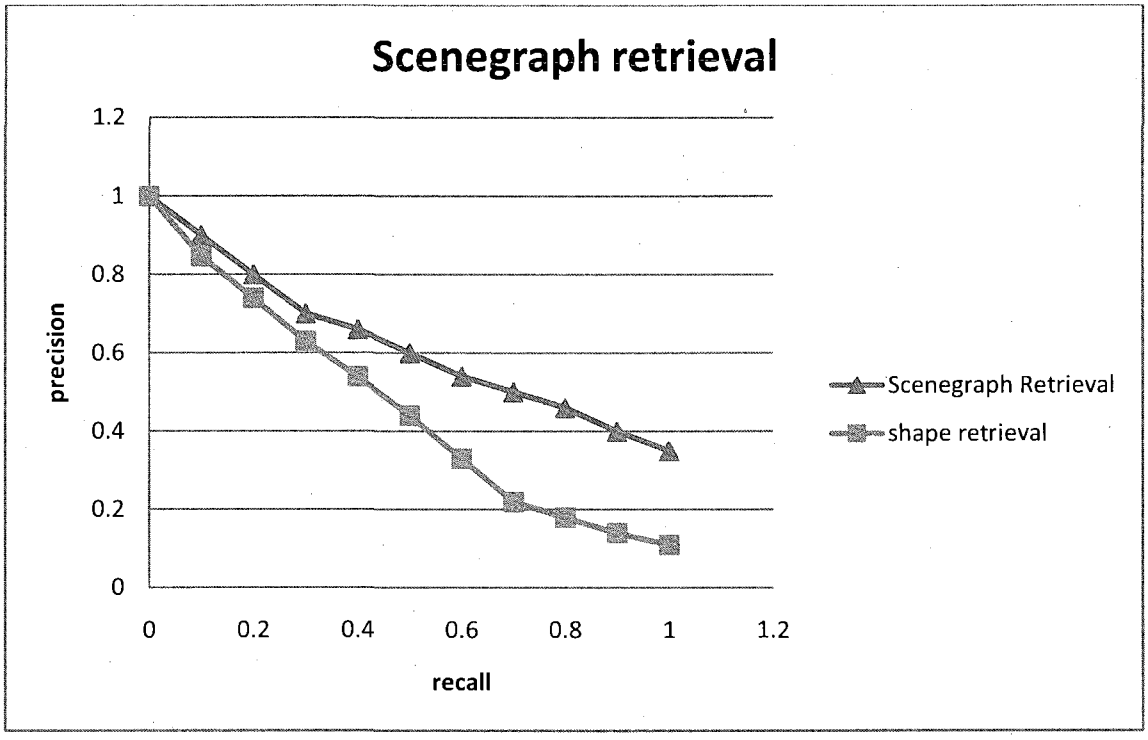


Figure 49: Scenegraph retrieval.

6.4.4 3D semantic retrieval

To test semantic retrieval, the TREE-D-SEEK corpus was used. Seven OWL ontology files were collected from the web. Corresponding to each file, ten OWL based semantic annotations files were created. Each of the ten semantic annotation file corresponding to the OWL ontology contain random individuals/instances corresponding the classes available in the ontology.

The recall/average precision graph obtained using the implemented semantic retrieval strategy is shown in Figure 50.

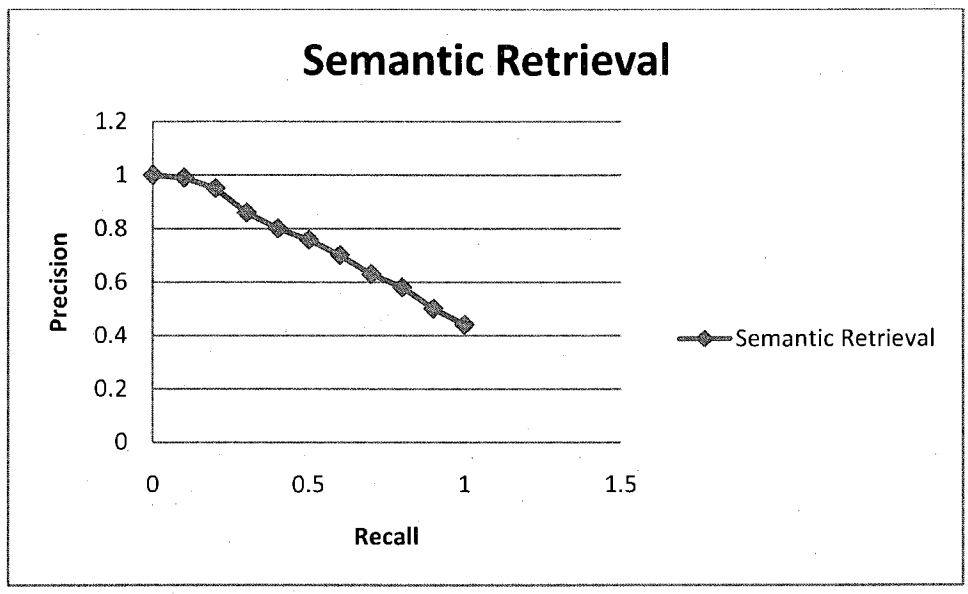


Figure 50: Average precision/recall for semantic retrieval.

6.5 Discussion

As the primary objective of this work is to design and implement a framework that is capable of retrieving 3D scenes and not necessarily to find or evaluate retrieval algorithms, these experiments result in three observations. First, performance metrics for retrieval may be collected using the TREE-D-SEEK framework. Second, the framework may be tested on several information sources and levels. Third, based on the recall-precision curves obtained for each type of retrieval, the behavior of the TREE-D-SEEK framework and its indexing algorithms is not inconsistent with what may be expected from IR systems.

Since the CSAM module is not fully implemented, it cannot be tested to show all the benefits described in the previous chapter. At the time of writing this dissertation, the CSAM module can only support basic scene based queries such as retrieving all objects in a scene that have the same material, color etc.

Based on the experiments conducted in this chapter, the usability of the framework or a search system that may use this framework may be theorized. A search system from the end user's perspective is shown in Figure 51.

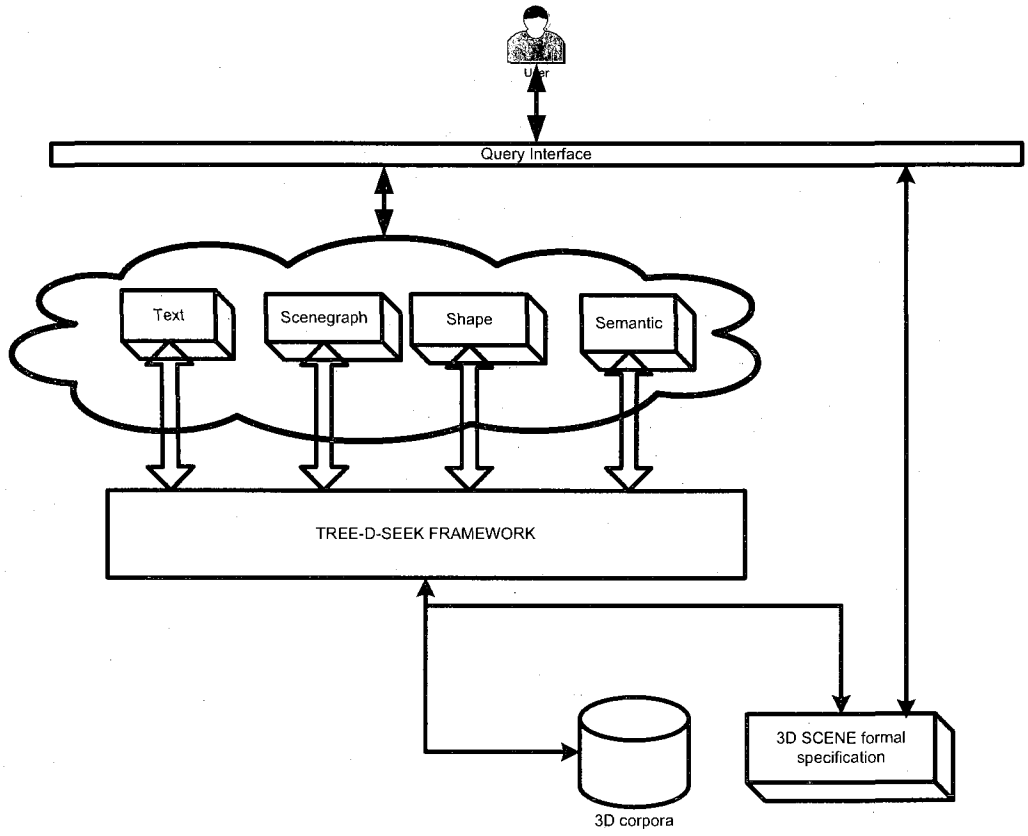


Figure 51: 3D search perspectives.

From the end-user's perspective, the TREE-D-SEEK framework can support retrieval of 3D scenes based on multiple "views" for indexing and searching. Each view presents aspects of the 3D scene at different information levels. This research suggests that an end user can indeed retrieve 3D content based on separating or combining views or by using a drill down strategy, wherein, for example, a user starts by performing a text search and this leads to an automated retrieval using the rest of the views. Multimodal queries may also be used by assigning numeric weights to each view. The appropriate querying strategy clearly will depend on several factors such as level of expertise of the end-user, type of information need, and etc. This research also highlights the importance of providing a search capability at two levels for 3D scenes. The first is at the document level wherein an entire scene may be returned as results of a search. The second level is to provide a search capability within a 3D scene. This is conceptually supported by the framework by using the TREE-D-SEEK semantic annotation model and the CSDF ontology.

Chapter VII

CONCLUSIONS AND FUTURE WORK

In summary, a strategy for retrieving 3D scenes is presented. The proposed strategy is to unify processes for retrieving 3D scenes from information sources. Relevant information on a 3D scene may be available as free text annotations and stored as text files. If a 3D scene is web-based, then the webpage containing the 3D scene, URL, 3D scene file name, etc. are potential information sources. If a 3D scene is scenegraph based then the scenegraph may also be an information source. A 3D scene may also contain low level content that may be used for indexing and matching. Semantic annotations of 3D scenes based on domain specific ontologies may also provide information that may be used for 3D scene retrieval.

In this research, TREE-D-SEEK, a framework for retrieving 3D scenes, is presented. The TREE-D-SEEK framework implements the retrieval strategy proposed in this dissertation. The framework is capable of retrieving 3D scenes by indexing and matching free text annotations, scenegraphs, shapes and semantic annotations. A software architecture for the framework is designed and implemented by first analyzing the individual process flows in retrieving 3D scenes based on each of the above mentioned types of information sources and then by generalizing and abstracting each of the individual process flows into a common process flow. The software architecture design uses a façade based pattern to encapsulate and hide the complexity of each step in the

common process flow. The TREE-D-SEEK framework is implemented on top of the Apache Lucene IR library. The software architecture of the TREE-D-SEEK framework presents clear interfaces that may be used for customizing and implementing retrieval algorithms. The TREE-D-SEEK framework can therefore be used as a test bench for evaluating 3D retrieval algorithms. As proof of concept, several indexing and matching algorithms are implemented.

In this research, a new semantic annotation model is also proposed for annotating and indexing 3D scenes. The semantic annotation model is ontology based. The ontology provides a formal conceptualization of a 3D scene obtained from the Common Scene Definition Framework.

7.1 Contribution to existing research on 3D retrieval

Based on the review of research in 3D retrieval, work prior to this research has not outlined a generic strategy for retrieving 3D content wherein a unified approach was used for indexing and matching content from different information sources and information levels. In this work, a unified strategy and framework is proposed for retrieving 3D content based on indexing low-level content, syntactic metadata, scenegraph content and structure, and semantic annotations.

Existing search systems are closed systems. A description of the software architecture of the search systems from a software class perspective is not available. The capability to reuse and fine tune the search system for a specific corpus or for implementing a new

retrieval scheme is not available for a 3D search engine developer. In this dissertation, TREE-D-SEEK, an open, extensible framework for retrieving 3D scenes, has been proposed and implemented. By using the TREE-D-SEEK framework, a developer may develop indexing and matching algorithms for the supported information sources and create a search system based on user needs and type of corpora being indexed. The developer can develop both unimodal and multimodal query modes that are most suitable for retrieving content from a targeted corpus. A novel approach proposed in this research is to support retrieval and therefore queries based on scenegraph content and structure. No previous work has proposed retrieval based on scenegraph structure.

Finally, a new 3D annotation model for indexing 3D scenes has been proposed in this dissertation. The annotation model is based on an ontology derived from a rapid prototyping framework. The annotation model supports the retrieval strategy proposed in this dissertation.

Although it may not be appropriate to compare a framework to complete search systems, a comparison of capabilities of the TREE-D-SEEK framework to the above mentioned 3D search engines and repositories is shown in Table 3. The checkmarks indicate existing capabilities already implemented and provided as part of the framework.

Table 3: Capability comparison.

Search Engine Queries	Princeton Search Engine	3DESS	Google 3D Warehouse	Ogden VI	ITI 3D Search	TREE-D-SEEK
Textual queries	✓		✓			✓
Query by shape	✓	✓		✓	✓	✓
Scenegraph						✓
Metadata	✓		✓		✓	✓
Semantic Metadata						✓
2D Sketch	✓	✓				
3D Sketch	✓	✓				

7.2 Future enhancements

Several enhancements may be envisioned. An immediate enhancement is to include new indexing and matching algorithms in the TREE-D-SEEK framework. Descriptors generated via these indexing methods may be evaluated using the TREE-D-SEEK framework. Currently, the framework is capable of retrieving only 3D objects. Behavior of the objects in 3D scenes is not indexed. Future work would support retrieval based on 3D object behavior. Indexing of 3D object behavior in 3D scenes would enable matching of 3D scenes based on animation.

From an implementation standpoint, indexing of corpora is sequential; i.e., if the information sources contain information at different information levels, the content is indexed based on the order of indexing strategies appearing in the indexer configuration

file. An improvement can be envisioned wherein a multithreaded indexing model may be used. In such a model, text indexing, scenegraph indexing, shape indexing and semantic indexing may be performed in parallel.

The TREE-D-SEEK framework can be extended to encapsulate the process of relevance feedback and learning algorithms so that the precision of returned results may be improved. Relevance feedback is an IR concept wherein the information need of a user is represented and refined by a search system based on the user's relevance assessment of the retrieved results obtained from an initial query.

The TREE-D-SEEK semantic annotation model presents several opportunities for future research. First, the CSDF ontology may be extended to provide a high level concept for each class available in the CSDF framework. The CSAM module discussed in Chapter IV may be fully implemented to enable translation of a CSDF 3D scene specification to a formal scene specification using the CSDF ontology. Any content indexed by the TREE-D-SEEK framework may be indexed, serialized and stored using RDF/XML. This would enable querying each 3D scene using technologies such as RQL and SPARQL.

A 3D authoring environment that uses both the CSDF framework and the TREE-D-SEEK framework to create 3D scenes can be built. This would enable the search capability to be part of the authoring and rapid prototyping environment. The underlying semantic annotation model may also assist in the creation and annotation of the 3D scenes.

REFERENCES

[1] Microsoft. "HealthVault Beta." Internet: http://www.healthvault.com/index.html?rmproc=true [Jan. 18, 2009].

[2] Google. "Official Google Blog: Google Health." Internet: http://googleblog.blogspot.com/2008/02/google-health-first-look.html,[Jan. 18, 2009].

[3] Linden Lab. "Second Life: Virtual worlds, avatars, 3D chat, online meetings." Internet: http://secondlife.com/ [Jan. 18, 2009].

[4] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. "The Princeton Shape Benchmark," in *Proc. SMI*, 2004. pp 167-178.

[5] P. Gorder."3DESS: a search engine enters the third dimension,"*Computing in Science and Engineering,* Vol. 6(6), 2004, pp. 4-7.

[6] Google. "3D Warehouse." Internet: http://sketchup.google.com/3dwarehouse/. [Jan 18 2009].

[7] M. Suzuki, "A web based retireval system for 3D polygonal models," in *20th NAFIPS International Conference (NAFIPS2001),*2001, pp. 2271-2276.

[8] Informatics and Telematics Institute. "3DSearch - Informatics and Telematics Insitute." Internet: http://3d-search.iti.gr/3DSearch [March 6, 2009].

[9] L. Ding, T. Finin, A. Joshi, R.Pan, R. Cost, Y. Peng, P. Reddivari, V. Doshi, J. Sachs. "SWOOGLE: A search and metadata engine for the Semantic Web," in *Proceedings of the Thirtheenth ACM Conference on Information and Knowledge Management*, 2004, pp. 652-659.

[10] Princeton 3D shape group. "3D search engine." Internet: http://shape.cs.princeton.edu/search.html. [March 19, 2009]

[11] G.Vasilakis, M. Pitikakis, M. Vavalis, C. Houstis. "A Semantic Based Search Engine for 3D Shapes : Design and Early Prototype Implementation," *Integration of Knowledge, Semantics and Digital Media Technology, EWIMT,* 2005, pp. 391- 397.

[12] O. Symonova, M. Dao, G.Ucelli, R. Amicis, "Ontology Based Shape Annotation and Retrieval,"*2nd International Workshop on Contexts and Ontologies: Theory, Practice and Applications*, 2006.

[13] P. Min. "A 3D model search engine." PhD Thesis, Princeton University. New Jersey, 2004.

[14] T. Funkhouser. "Overview of 3D Object Representations."Internet: http://www.cs.princeton.edu/courses/archive/spr04/cos426/lectures/13-reps.pdf. [Dec. 1, 2008]

[15] H. Anan."Digital Library Services for 3D models." Phd Dissertation, Old Dominion University, Virginia , 2004.

[16] University of Iowa. "Voxel Processing in a nutshell." Internet: http://www.uiowa.edu/~image/iaf/concepts/voxels/voxel_processing.html [Dec. 1, 2008].

[17]. T. Funkhouser. "Scene Graphs and Modeling Transformations."Internet: http://www.cs.princeton.edu/courses/archive/fall00/cs426/lectures/transform/transform.pdf. [Dec 1, 2008.]

[18] L. Belfore II. "Virtual worlds: An architecture for constructing large VRML worlds." *Transactions of the Society for Computer Simulation International*, Vol. 18, pp. 24-40, 2001.

[19] W3C consortium. "X3D." Internet: http://www.web3d.org/about/overview/ [Feb. 5, 2009].

[20] D. Brutzman and D. Leonard. *X3D: Extensible 3D Graphics for web authors,* Morgan Kaufmann Publishers, 2007.

[21] MIT. "Scenegraphs." Internet: http://web.mit.edu/ivlib/www/iv/scenes.html [Feb 1, 2009].

[22] Java.net. "An introduction to scenegraphs." Internet: http://download.java.net/media/java3d/javadoc/1.4.0/javax/media/j3d/doc-files/SceneGraphOverview.html [February 4, 2009].

[23] E. Greengrass. "Coursework Materials fo IST 441 Information Retrieval and Search Engines." Internet: http://clgiles.ist.psu.edu/IST441/materials/texts/IR.report.120600.book.pdf [December 4, 2008]

[24] M: Damashek. "Gauging similarity with n-grams: Language-independent categorization of text." *Science.* Vol. 267, 1995, pp. 843-848.

[25]W. Cooper, F. Gey and D. Dabney. "Probabilistic retrieval based on staged logistic regression," in Proceesings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992, pp. 198-209.

[26] J. Hunt, K. Vo, W. Tichy. "An emperical study of delta algorithms." *Lecture notes in Computer Science.*Springer Berlin/Heidelberg, Vol. 1167, 1996, pp. 49-66.

[27] D. Trinkle. "Official RCS Homepage." Internet: http://www.cs.purdue.edu/homes/trinkle/RCS/ [December 25, 2008.]

[28] S. Chawathe, A. Rajaraman, H. Molina and J. Widom. "Change detection in heiracchically structured information," in Proceedings of the ACM SIGMOD International Conference on Management of Data, 1996., pp. 493-504.

[29] G. Valiente. *Algorithms on trees and graphs.* Springer-Verlag, 2002.

[30] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms,* Addison-Wesley, 1974.

[31] K. Tai. "The tree to tree correction problem." *Journal of the Association for Computing Machinery,* Vol. 26, pp. 422-433,1979.

[32] S. Selkow. "The tree to tree editing problem", Information Processing Letters, pp. 184-186, vol 6, 1977.

[33] K. Zhang, R. Statman, and D. Shasha. "On the editing distance between unordered labeled trees." Information Processing Letters, vol. 42, pp. 133-139, 1992.

[34] G. Valiente. Simple and efficient subtree isomorphism. Department of Software, Technical University of Catalonia. 2000. LS1-00-72-R.

[35] G. Valiente. *Simple and efficient tree comparison.*Technical report, Department of Software, Technical University of Catalonia. 2001.

[36] G. Cobena, S. Abiteboul, and A. Marian, "Detecting changes in XML documents," in *Proceedings of the 18th International Conference on Data Engineering*, 2002, pp. 41-52.

[37] W. Yuan,D. Dewitt, J. Cai. "X-diff: An effecive change detection algorithm for XML documents," in Proceedings of the 19th International Conference on Data Engineering, 2003, pp 519-530.

[38] F.Cubera and D. Epstein.*"Fast difference and Update of XML documents," in* XTech, 1999.

[39] H. Mayurama, K. Tamura and R.Uramoto. "RFC-2803. Dom-Hash." Internet: http://tools.ietf.org/html/rfc2803 [December 28, 2008]

[40] JohanTangelder, and Remco Vetcamp. "A survey of content based 3D shape retrieval methods, "in Proc. SMI, 2004, pp. 144-156.

[41] C.Zhang, C. and T. Chen. "Efficient feature extraction for 2D/3D objects in mesh representation."*in* CIP, 2001, 935-938.

[42] E. Paquet. "Description of shape information for 2D and 3D objects." *Signal Processing: Image Communication*,vol. 16, pp. 103-122, 2000.

[43] R. Osada,T. Funkhouser,B. Chazelle, D. Dobkin."Shape distributions."*ACM Transactions on Graphics*, vol 21, pp. 807-832. 2002.

[44] M. Ankerst, G. Kastenmüller, H. Kriegel, T. Seidl. "3D shape histograms for similarity search ad classification in spatial databases." *Lecture Notes in Computer Science(SSD),* vol 1651, pp. 207-226, 1999.

[45] M.Kazdhan, T.Funkhouser and S. Rusinkiewicz. "Rotational invariant spherical harmonic representation of 3D shape descriptor," *Symposium on Geometry Processing,* 2003.

[46] M.Novotni, M. and R. Klien. "3D Zernike descriptors for content based shape retrieval." *Solid Modeling*, pp. 216-225, 2003.

[47] H. Shum, M. Hebert, and K. Ikeuchi. "On 3D shape similarity," *Proceedings IEEE Computer Vision and Pattern Recognition.* 1996. pp. 526-531.

[48] D. Messerschmitt and H. Varian. "IS223: Strategic Computing and Communications Technology." Internet: http://www2.sims.berkeley.edu/courses/is224/s99/GroupG/report1.html [December 15, 2008]

[49] F. Pereira and R. VandeWalle. "Multimedia Content Description in MPEG-7 and MPEG-21," in *Multimedia Content and the Semantic Web: Standards, Methods and Tools*, G. Stamou and S.Kollias, Ed., John Wiley & Sons Ltd., 2005, pp. 3-41.

[50] T. Gruber. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing." *International Journal Human-Compute Studies*, Vol. 43, pp. 907-928, 1995.

[51] T. Berners-Lee. "WWW past and future." http://www.w3.org/2003/Talks/0922-rsoc-tbl/slide30-0.html [Dec. 20, 2008]

[52] P. Karp, V. Chaudri, and J. Tomere. " XOL: An XML Based Ontology Exchange Language." Internet: http://www.ai.sri.com/pkarp/xol/xol.html [Jan 6, 2009]

[53] S. Luke, D. Rager and J. Hendler. " Ontology-based Web agents,"*ACM International Conference on Autonomous Agents,* ,1997. pp. 59-66.

[54] Ontologos. "Ontology Markup Language." Internet: http://www.ontologos.org/OML/OML%200.3.htm [Cited: Jan 6, 2009]

[55] W3C. "Resourse Description Framework." Internet: http://www.w3.org/RDF/ [January 6, 2009]

[56] W3C. "RDF Schema." Internet: http://www.w3.org/TR/rdf-schema/ [Jan 6, 2009]

[57] Ontoknowledge. "OIL page." Internet: http://www.ontoknowledge.org/oil/ [Jan. 6, 2009]

[58] DARPA. "DARPA Agent Markup Language." Internet: http://www.ontoknowledge.org/oil/ [Jan. 6, 2009]

[59] W3C. "OWL Web Ontology Language." Internet: http://www.w3.org/TR/owl-features/ [Jan. 6, 2009]

[60] N. Noy and M. Musen. "PROMPTDIFF: A fixed-point algorithm for comparing ontology versions,"Eighteenth National Conference on Artificial Intelligence. pp. 744-750.

[61] N. Noy and M. Musen. "Anchor-PROMPT: Using non-local context for semantic matching," in *Proceedings of the workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence*. 2001. pp. 63-70.

[62] M. Klein, D. Fensel, A. Kiryakov and D. Ognyanov. " Ontology versioning and change detection on the Web," *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02),* 2002, pp. 197-212.

[63] A. Doan, P. Domingos, and A. Halevy. "Reconciling Schemas of Disparate Data Sources," *in proceedings of the ACM SIGMOD Conference on Management of Data* (SIGMOD-2001), 2001, pp. 509-520.

[64] P. Halabala. "Semantic Metadata Creation," *7th Central European Seminar on Computer Graphics (CESCG 2003).* 2003, pp. 15-25.

[65] H. Mansouri. *Using Semantic Descriptions for Building and Querying Virtual Environments.* PhD thesis , Vrije Universiteit. Brussels, 2005.

[66] VRWISE. "OntoWorld tool." http://wise.vub.ac.be/ontobasis/tools.html. [Jan 6, 2009]

[67] I.Bilasco,J. Gensel, M. Oliver and H. Martin. "On Indexing of 3D scenes using MPEG-7,"*13th annual ACM international conference on Multimedia*. 2005. pp. 471 - 474 .

[68]. *Semantic Description of 3D Enviroments: a Proposal Based on Web Standards.* 2006 . 11th international conference on 3D web technology. pp. 85 - 95 .

[69] F. Pitarello, A. Faven, E. Comossi, F. Giannini and M.Monti. "Deriving Functionality from 3D Shapes: Ontology Driven Annotation and Retrieval.", *Computer-Aided Design & Applications*, vol. 4, pp. 773-782, 2007.

[70] T. Funkhouser, P. Min, M. Kazdhan, J. Chen, A. Halderman, and D. Dobkin, "A search engine for 3D models." *ACM transactions on Graphics,* vol. 22, pp. 83-105, 2003.

[71]Google. "Google Image Search." Internet: http://images.google.com/ [Cited: Jan. 6, 2009]

[72] Emre Baydogan, Rapid Prototyping for Virtual Environments. PhD thesis, Old Dominion University, Virginia, Dec. 2008.

[73] P. Min, M. Kazdhan, T. Funkhouser. "A comparison of text and shape matching for retreival of online 3D models," in *Proceedings European Conference on Digital Libraries.*2004, pp. 209-240.

[74] W. Bille, B. Pellens, F. Kleinermann, "Intelligent Modelling of Virtual Worlds Using Domain Ontologies." *Workshop of Intelligent Computing (WIC)*,2004, pp. 272-279.

[75] E. Kalogerakis, C. Stavros, N. Moumoutzis. "Coupling Ontologies with Graphics Content for Knowledge Driven Visualization," *IEEE Virtual Reality Conference*, 2006, pp. 43-50.

[76] G. Pirro and D.Talia. "An approach to Ontology mapping based on the Lucene search engine library," in *18th Internation Workshop on Database and Expert Systems Architecture,* 2007, pp. 407-411.

[77] Apache Lucene. "Apache Lucene Overview." Internet: http://lucene.apache.org/java/docs/ [Jan. 15, 2009]

[78] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software,* Addision Wesley, 1995.

[79] J. Nie, and Fuman Jin. "A multilingual approach to multilingual Information Retrieval." *Lecture Notes In Computer* Science, Springer Berlin / Heidelberg, vol. 2785, 2003.

[80] M. Porter. "An algorithm for suffix stripping."*Program*, vol. 14, pp. 130-137, 1980.

[81] C. Fellbaum. *Wordnet: An Electronic Lexical Database.* 1998.

[82] J.Wang, Y. He, and H. Tian. "Voxel-based shape analysis and search of mechanical CAD-models.", *Forschung im Ingenieurwesen*, vol. 71, pp. 185-195, 2007

[83] P. Min. "Binvox." Internet: http://www.google.com/search?q=binvox [Nov 12, 2008] .

[84] P. Min. "Thinvox." Internet: http://www.google.com/search?q=thinvox [Nov 12, 2008].

[85] Stanford Center for Biomedical Informatics Research ,Stanford University. " Protege Ontology Editor." Internet: http://protege.stanford.edu/ [Dec 12, 2008].

[86] University of Manchester;Clark & Parsia LLC,University of Ulm. "OWL API." Internet: http://owlapi.sourceforge.net/ [Nov. 11, 2008].

[87] A. Ames, D. Nadeau, J. Moreland. "The VRML2.0 Source book." Internet: http://www.wiley.com/legacy/compbooks/vrml2sbk/toc/toc.htm [Dec. 25, 2008].

[88]3D Cafe. "3D Cafe -Home." Internet: http://www.3dcafe.com/index.php?option=com_frontpage&Itemid=1 [Dec. 21, 2008].

[89] Oxford University. "Oxford Text Archive." Internet: http://ota.ahds.ac.uk/catalogue/index-id.html [March 17, 2009].

[90] T. Saracevic. "Evaluation of Evaluation in Information Retrieval," in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.

[91] T. Saracevic. "RELEVANCE: A review of and a framework for the thinking on the notion in Information Science."*Journal for the American Society for Information Science*, vol 6, pp. 321-343, 1995.

APPENDICES

A.1 Relevance and Information Retrieval (IR) performance metrics

Most IR performance metrics are based on the metrics of precision and recall. *Precision* is defined as the "the ratio of relevant items retrieved to all items retrieved or the probability that given that an item is retrieved, it is relevant" [90]. *Recall* is defined as the "ratio of relevant items retrieved to all relevant items in a file [collection] or given the probability that an item is relevant, it will be retrieved" [90]. Relevance is an important criterion in the evaluation of IR systems. Several IR experts [91] have partitioned relevance into two categories: objective or system based relevance and subjective or human based relevance. Objective or system based relevance is an algorithmic process wherein the query is matched with the contents of the document without heeding the context of the query. Subjective relevance deals with relevance based on the erudition level of the user. Subjective relevance is broken into four further subcategories namely *generic topicality*, *pertinence*, *situational*, *motivational* and *affective* relevance.

A.1.1 Recall and precision

Recall and precision are defined in (7) and (8) below. Recall provides a measure of the effectiveness of a search system in retrieving all relevant documents from a corpus. Precision provides a measure of the effectiveness of a search system in retrieving the most relevant documents from a corpus.

$$recall = \frac{\text{Number of retrieved relevant documents}}{\text{Total number of relevant docuement}} \tag{7}$$

$$precision = \frac{\text{Number of retrieved relevant documents}}{\text{Total number of retrieved docuement}} \quad (8)$$

Precision and recall are competing metrics. Often in IR systems, if the recall level is high, the precision of the returned results is low or vice versa. To control the two metrics, a threshold value may be used that indicates the user-defined cutoff similarity value.

For the purpose of this research, recall and precision may be calculated as follows. A given corpus must be classified into categories apriori. Then, a document is matched with every document (including itself) in the corpus. Each document in the corpus can then be ranked based on the score obtained by use of the matching algorithm. The retrieved documents may belong to one of the classified categories in the corpus. Then, assuming the class size is x, the recall value for a document from the same category going down the ranked list will be 1/x , 2/x, … to x/x. Then, a precision value corresponding to this recall value will be the number of relevant documents divided by the number of results it took to get to that number. The number of results it took to get to a particular result is also equal to the rank of the document. In Table 4, an example is provided to shown how precision recall values are calculated for a particular query. In this example, assume that there were ten relevant documents in the corpus. For the particular query, twenty documents have been retrieved. The leftmost column shows the ordering of documents returned from the search. The second column contains the relevance of the document to the query. It is a Boolean value. Recall is calculated based on (7) and (8).

Table 4: Example of PR calculation.

RANK	RELEVANCE	RECALL	PRECISION
1	RELEVANT	10%	100%
2	RELEVANT	20%	100%
3	RELEVANT	30%	100%
4	NOT	30%	75%
5	RELEVANT	40%	80%
6	NOT	40%	67%
7	RELEVANT	50%	71%
8	NOT	50%	63%
9	NOT	50%	55%
10	RELEVANT	60%	60%
11	NOT	60%	55%
12	NOT	60%	50%
13	RELEVANT	70%	54%
14	NOT	80%	50%
15	NOT	80%	47%
16	RELEVANT	90%	50%
17	NOT	90%	47%
18	RELEVANT	100%	50%
19	RELEVANT	100%	53%
20	NOT	100%	50%

Standardized recall-precision values are shown in Table 4. In the leftmost column, the standard 11 recall points are shown. In the second column, the corresponding highest precision value from Table 4 for a particular standard recall value is used. This value indicates the maximum number of documents required to achieve that recall level. The rightmost column depicts the interpolated precision. Interpolated precision is the maximum precision value taking into account precision at the current recall level and precision at subsequent recall levels.

Table 5: Interpolated precision.

Recall	Precision	Interpolated precision
0%	-	100%
10%	100%	100%
20%	100%	100%
30%	100%	100%
40%	80%	80%
50%	71%	71%
60%	60%	60%
70%	54%	54%
80%	50%	53%
90%	50%	53%
100%	53%	53%

For 11 recall intervals, precision values are averaged over all queries. If a single value is desired, the 11 point average may be calculated. For Table 5, the 11 point average precision is approximately 75 percent.

VITA

DEGREES:

- Doctor of Philosophy (Electrical and Computer Engineering) , Old Dominion University, Norfolk, VA, May 2009
- Master of Science (Computer Engineering), Old Dominion University, Norfolk, VA, December 2002
- Bachelor of Engineering (Computer Science and Engineering), PES Institute of Technology, Bangalore University, Bangalore, India, October 1999

PART TIME EMPLOYMENT:

- Software engineering intern at Simulation and Training Environment Lab (SiTEL), October 2008 –Present
- Research assistant at the Virginia Modeling and Simulation Center (VMASC), January 2001- May 2008

PUBLICATIONS

Book Chapters

- J. Leathrum, R. Mielke, S. Mazumdar, R. Mathew, Y. Manepalli, V. Pillai and R. Malladi. "A Simulation Architecture to Support Intratheater Sealift Operations." *Defense Transportation: Algorithms, Models and Applications for the 21st Century*, Elsevier Science Ltd, 2004.

Journal Articles

- R. Mathew, J. Leathrum, S. Mazumdar, T. Frith, J. Joines. "An Object-Oriented Architecture for the Simulation of Networks of Cargo Terminal Operations." *Journal of Defense Modeling and Simulation (JDMS),* vol. 2, pp. 101-116.2005.
- J. Leathrum., R. Mielke, S. Mazumdar, R. Mathew, Y. Manepalli, V. Pillai and R. Malladi. "A Simulation Architecture to Support Intratheater Sealift Operations."*Mathematical and Computer Modelling,* vol.39, 817–838, May 2004.

Conference Papers

- E. Baydogan, S. Mazumdar, L. Belfore. "Decoupled Agent Architecture for Virtual Operating Room Training Simulations", *CGVR '08: Proceedings of the 2008, International Conference on Computer Graphics and Virtual Reality, 2008.*
- E. Baydogan, S. Mazumdar, L. Belfore, "Simulation Architecture for Virtual Operating Room Training", *2008 VMASC Capstone Conference, 2008.*
- S. Mazumdar, E. Baydogan, L. Belfore. "OntoVOR: The Design of a Knowledge-base for a Virtual Operating Room," *Proceedings of the 2007 ModSim World Conference*, 2007.
- L. Belfore, S. Mazumdar, S. Rizvi, J. Garcia, D. Bitts, C. Blancett, E.Paredes, D. Moulton, W. Quinones, K. Jones, Jennifer Browning. "Integrating the Joint Operations Feasibility Tool (JOFT) with JFAST, " in *Proceedings of SIW Workshop, July 2006. Proceedings of SIW Workshop*, July 2006.